Jakob Jünger
Chantal Gärtner



Computational Methods für die Sozial- und Geisteswissenschaften





Computational Methods für die Sozial- und Geisteswissenschaften

Jakob Jünger • Chantal Gärtner

Computational Methods für die Sozial- und Geisteswissenschaften



Jakob Jünger
Universität Münster
Münster, Deutschland

Chantal Gärtner
Universität Münster
Münster, Deutschland



ISBN 978-3-658-37746-5 ISBN 978-3-658-37747-2 (eBook)
https://doi.org/10.1007/978-3-658-37747-2

Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über http://dnb.d-nb.de abrufbar.

Springer VS

Planung/Lektorat: Barbara Emig-Roller
Springer VS ist ein Imprint der eingetragenen Gesellschaft Springer Fachmedien Wiesbaden GmbH und ist ein Teil von Springer Nature.
Die Anschrift der Gesellschaft ist: Abraham-Lincoln-Str. 46, 65189 Wiesbaden, Germany

Präambel

Dieses Buch ist das Ergebnis vieler glücklicher Umstände. Als wir am Anfang des Studiums begannen, uns mit sozial- und geisteswissenschaftlichen Themen zu beschäftigen, haben wir nicht kommen sehen, wie sich unsere Begeisterung für Computation damit verbinden wird. Lange Zeit gab es noch keinen etablierten Namen für die Verknüpfung zwischen Informatik auf der einen Seite und Sozial- und Geisteswissenschaften auf der anderen. Und auch wenn die Bezeichnungen bis heute volatil sind, ist klar geworden: Computational Methods haben sich im Feld etabliert. Advokat:innen und Evangelist:innen haben an Begriffsverständnissen gearbeitet und sich mit epistemologischen Fragen beschäftigt. Wissenschaftsmanager:innen haben Aufbauarbeit geleistet, um technische und organisatorische Infrastrukturen zu schaffen. Methodenentwickler:innen haben Tools und Modelle entwickelt, deren Brauchbarkeit für den Erkenntnisgewinn von Anwender:innen evaluiert werden. Digital Humanties und Computational Social Sciences sind in einigen Curricula von Studiengängen verankert. Aus vagen Intuitionen ist Wirklichkeit geworden.

Wir sind dankbar für die Menschen und die Institutionen, die uns auf unserem Weg begleitet haben und die wir begleiten durften. An der Universität Greifswald hat sich nicht nur die Finanzierung für dieses Buch gefunden, sie hat Freiräume für unsere persönliche kreative Entwicklung geschaffen. Die Universität Münster und die Akademie der Wissenschaften und der Literatur Mainz haben ein Umfeld geschaffen, in dem wir das Werk gemeinsam fortführen konnten. Für die Möglichkeit zur Publikation danken wir dem Springer VS Verlag, namentlich Barbara Emig-Roller und Britta Laufer. Korrekturlesend hat sich dankenswerterweise Dagmar Schierenberg durch das Manuskript gearbeitet. An Emilie Hemmerling geht der Dank für die Kapitelbilder. Es ist nicht selbstverständlich, dass wichtige strukturelle Voraussetzungen von Personen und Einrichtungen geschaffen werden, wenn sie selbst kaum Berührungspunkte mit einem Thema haben.

Ganz besonders wichtig waren die Studierenden, die sich, ohne zu wissen, was sie erwartet, in verschiedenen Seminaren auf die Welt automatisierter Verfahren eingelassen haben. In der Regel ohne informatischen Hintergrund sind sie mit uns bei guter Laune an die Grenzen des Verständnisses und darüber hinaus gegangen. Einige dieser Studierenden sind direkt durch Recherchen, Tests und Korrekturen an diesem Buch beteiligt, ihnen gilt besonderer Dank: Silja Klein, Julia Alili, Pauline Haß, Isabel Kleefeld, Vicky Wagemann und Henrieke Kotthoff.

Viele Kapitel des Buches sind als kurze Tutorials entstanden – für die daraus resultierende Brüchigkeit des Textes übernehmen wir gern die Verantwortung. Die Idee bestand stets darin, Türen zu komplexen Themengebieten zu öffnen, indem man innerhalb begrenzter verfügbarer Zeit erste Erfahrungen sammelt und dabei gefahrlos stolpern kann. Wir hoffen, einige unserer Erfahrungen nun weitergeben zu können und wünschen viel Spaß bei den wohl wichtigsten Tätigkeiten im Bereich der Computational Methods: dem Fehlermachen, Fehlerfinden und Fehlerbeheben.

Münster & München, Deutschland — Jakob Jünger
Frühling 2022 — Chantal Gärtner

Inhaltsverzeichnis

Teil I Grundlagen

1 Einleitung und Überblick 3
1.1 Was sind Computational Methods? 6
1.2 Der Werkzeugkoffer 13
1.3 Mit Fehlern umgehen 21
1.4 Überblick über das Buch 23
Literatur 27

2 Datenquellen 31
2.1 Webseiten 34
2.2 Application Programming Interfaces 37
2.3 Datenbanken und Datensätze 42
Literatur 48

3 Datenformate 53
3.1 Datentypen 55
3.2 Textformate (MD) 58
3.3 Tabellenformate (CSV) 61
3.4 Auszeichnungssprachen (HTML und XML) 62
3.5 Objektdatenformate (JSON) 67
3.6 Datenbanken (SQL und NoSQL) 68
3.7 Datenmodelle (RDF) 72
Literatur 79

4 Datenextraktion . . . 83
4.1 Selektionsverfahren und -sprachen . . . 85
4.2 Transformationsverfahren . . . 112
Literatur . . . 128

Teil II Programmierkonzepte

5 Programmierung . . . 133
5.1 Einführung in die Datenanalyse mit R . . . 136
5.2 Einführung in die Datenanalyse mit Python . . . 174
Literatur . . . 202

6 Scaling up. Daten und Skripte organisieren . . . 207
6.1 Versionsverwaltung . . . 209
6.2 Entwicklungsumgebungen . . . 214
6.3 Laufzeitumgebungen und virtuelle Boxen . . . 220
6.4 Parallelisierung mit Cores, Clustern und Cloud Computing . . . 235
Literatur . . . 249

Teil III Anwendungsfelder

7 Automatisierte Datenerhebung . . . 255
7.1 Webscraping . . . 257
7.2 Application Programming Interfaces (APIs) . . . 282
Literatur . . . 301

8 Maschinelles Lernen . . . 305
8.1 Überwachte Lernverfahren . . . 308
8.2 Unüberwachte Lernverfahren . . . 336
Literatur . . . 350

9 Textanalyse . . . 355
9.1 Wortfrequenzanalysen . . . 357
9.2 Diktionärbasierte Inhaltsanalyse . . . 373
9.3 Syntax und Semantik . . . 380
Literatur . . . 386

10 Netzwerkanalyse . . . 389
10.1 Grundlegende Konzepte der Netzwerkanalyse . . . 392
10.2 Erhebung, Analyse und Visualisierung von Netzwerken . . . 402
Literatur . . . 419

11 Simulationsverfahren 423
11.1 Eine Welt entsteht 428
11.2 Die Welt entwickelt sich 429
11.3 Analyse der Ergebnisse 431
Literatur 433

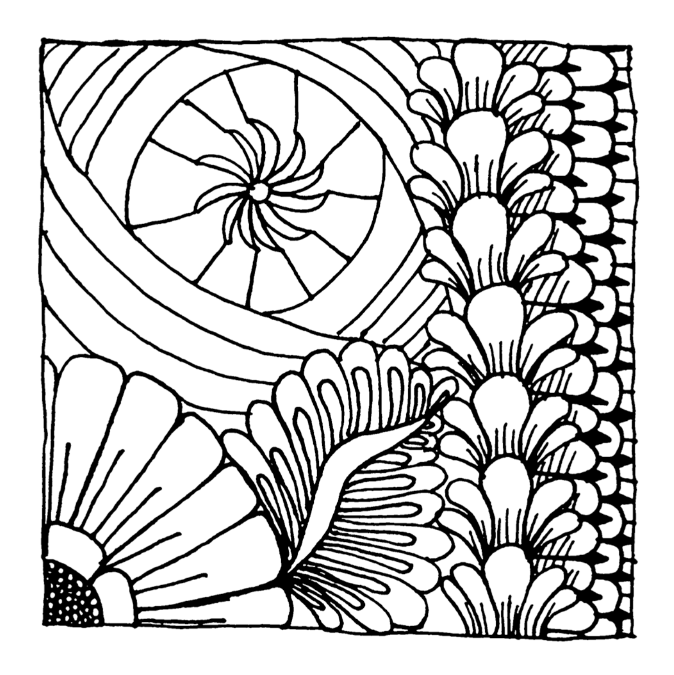
12 Zusammenfassung 435
Literatur 438

Literatur 439

Teil I

Grundlagen

1 Einleitung und Überblick



J. Jünger, C. Gärtner, *Computational Methods für die Sozial- und Geisteswissenschaften*, https://doi.org/10.1007/978-3-658-37747-2_1

Zusammenfassung

Dieses Kapitel führt Sie in die Welt der Computational Methods ein. Sie erfahren, an wen sich das Buch richtet, was unter Computational Methods verstanden wird und wofür diese in der Wissenschaft eingesetzt werden. Zudem lernen Sie, welche grundlegenden Tools für die Arbeit mit diesen Methoden benötigt werden.

Im Online-Repositorium unter https://github.com/strohne/cm finden Sie begleitend zum Kapitel weitere Materialien, auf die wir im Text mit ☛ verweisen.

Schlüsselwörter

Computational Methods · Computational Social Science · Digital Humanities · Kommandozeile · Texteditor

Der Bereich Computational Methods hat für die Sozial- und Geisteswissenschaften zunehmend an Bedeutung gewonnen. Das kann unter anderem darauf zurückgeführt werden, dass Kulturgüter wie historische Bücher, Daten von Organisationen oder auch die Kommunikation in journalistischen Medien in den letzten Jahrzehnten digital verfügbar wurden. Nicht nur das: Viele soziale, auch interpersonale, Prozesse laufen mittlerweile zum Beispiel auf Online-Plattformen direkt in digitalen Umgebungen ab. Computational Methods helfen dabei, umfangreiche und heterogene Datenbestände zusammenzustellen und automatisiert auszuwerten. Daraus ergeben sich auch weitere wissenschaftliche Fragestellungen und Herangehensweisen, indem beispielsweise menschliches Verhalten mit Simulationsmodellen nachgestellt wird, um aus dem Vergleich mit der sozialen Wirklichkeit grundlegende Mechanismen herauszuarbeiten. Besonders reizvoll ist dabei, dass gleichzeitig eine Vogelperspektive und eine Froschperspektive eingenommen werden können. So kann beispielsweise mit Netzwerkanalysen die Struktur hinter dem Zusammenspiel einer Vielzahl an Akteuren[1] visualisiert werden, ohne dass die einzelnen Akteure aus dem Blick geraten müssen.

Anzeichen für die zunehmende Bedeutung von Computational Methods sind auch die Gründung von spezialisierten Fachzeitschriften, Forschungseinrichtungen

[1] Sofern wir davon ausgehen, dass es sich um komplexe nichtmenschliche Akteure handelt (korporative oder kollektive Akteure sowie technische Rollen), gendern wir in diesem Buch Begriffe wie Akteur oder Anbieter nicht, auch um eine Anthropomorphisierung zu vermeiden.

und Lehrstühlen sowie Fachvertretungen. In der Kommunikationswissenschaft wird dies beispielsweise an der Gründung einer Computational Methods Division innerhalb der internationalen Fachvertretung International Communication Association (ICA) deutlich. Ausgangspunkt war die Idee, dass Computational Methods eine wichtige Rolle für die Weiterentwicklung kommunikationswissenschaftlicher Forschung spielen werden: „We believe that computational methods will be at the forefront of progress in the field in the coming decades“ (ICA CM 2017). Damit aber Computational Methods im Rahmen substanzieller Forschung eingesetzt werden können, braucht es Wissenschaftler:innen, die mit diesen Methoden umgehen können.

Das vorliegende Buch soll (bisherige) technische Laien in die Lage versetzen, selbstständig automatisierte Verfahren der Datenerhebung und Datenauswertung anzuwenden. Es ist begleitend zu Lehrveranstaltungen entstanden, in denen für Studierende der Sozial- und Geisteswissenschaften der Weg in eine zunächst fremde, dann aber sehr spannende Welt eröffnet wird. Dabei geht es darum, diese digitale Welt nicht nur besser zu verstehen, sondern auch mitgestalten zu können.

Im ersten Teil des Buches wird grundlegendes Wissen über Datenquellen, Datenformate und Verfahren zur Aufbereitung von Daten vermittelt. Im zweiten Teil finden sich kurze Einführungen in die zwei einschlägigen Programmiersprachen R und Python sowie in Konzepte zum Verwalten größerer Programmierprojekte. Im dritten Teil werden spezielle Erhebungs- und Auswertungsverfahren vorgestellt, beispielsweise Webscraping, Textanalyse und Klassifikationsverfahren. Innerhalb der Kapitel wird neben einem Überblick über die Verfahren jeweils eine Anleitung für ein ausgewähltes Beispiel gegeben. Die Beispiele beziehen sich häufig auf kommunikationswissenschaftlich relevante Bereiche, verbunden mit der Hoffnung, dass die Konzepte und Anleitungen anschlussfähig für sehr unterschiedliche Felder sind. Denn die Kommunikationswissenschaft ist eine integrative Disziplin, in der sowohl sozial- als auch geisteswissenschaftliches Denken zusammenkommt.

Die vorliegende Einführung gibt einen ersten Einblick in die vorgestellten Bereiche, kann aber nicht die vielen, sehr guten Einführungswerke in spezialisierte Verfahren ersetzen. Deshalb sind am Ende jedes Kapitels weiterführende Literaturhinweise und themenspezifische Einführungswerke zusammengestellt. Daneben finden sich nach jedem Kapitel Übungsfragen, mit denen theoretisches und praktisches Wissen überprüft werden kann. Zusätzlich zu den Inhalten in diesem Buch werden Daten und Skripte in einem Repositorium als Begleitmaterialien zur Verfügung gestellt. Wie diese beim Lernen von Computational Methods helfen können, wird in Abschn. 1.2.4 beschrieben.

1.1 Was sind Computational Methods?

Ausgehend von der Bezeichnung „Computational Methods“ wären darunter alle Verfahren zu verstehen, bei denen „Computation“ zur wissenschaftlichen Analyse eingesetzt wird. Versteht man unter dem Begriff „Computation“ in direkter Übersetzung allerdings ganz allgemein jede Form des Rechnens, dann umfasst das auch die quantitative Auswertung von Befragungen und Texten und jede Art von Statistik, im einfachsten Fall sogar die Bildung von Summen und Mittelwerten. Auch eine Eingrenzung auf *maschinelles* Rechnen hilft hier nicht weiter, denn Maschinen sind im Verständnis der Informatik nicht zwangsläufig elektronische oder mechanische Geräte. Ein Beispiel für eine rein konzeptionelle Maschine ist die Turing-Maschine, die ein Rechenmodell bzw. die mathematische Version eines Algorithmus darstellt. Zudem werden auch in der klassischen statistischen Analyse Computer eingesetzt. Im Film Hidden Figures (2016) findet sich ein anschauliches Beispiel dafür, dass unter Computern auch Menschen mit einer Begabung für das Rechnen verstanden werden können. Die „Computational Unit“, in der Berechnungen für die Flugbahnen der Apollo-Raumfahrtmission durchgeführt werden, besteht zunächst ausschließlich aus Frauen. Erst am Ende des Films wird dargestellt, wie diese Aufgabe von elektronischen Maschinen übernommen wird.

Es ist aber gerade dieser Aspekt, das Hinausgehen über klassische sozial- und geisteswissenschaftliche Methoden, der als Besonderheit von Computational Methods angesehen wird (siehe auch van Atteveldt und Peng 2018, S. 82). Einigkeit besteht in den Sozialwissenschaften zumindest dahingehend, dass es sich eben nicht um klassische Methoden handelt, sondern um Verfahren aus der Informatik: „Computational social sciences is a research discipline at the interface between computer science and the traditional social sciences. This interdisciplinary and emerging scientific field uses computationally methods to analyze and model social phenomena, social structures, and collective behavior“ (Amaral 2017; siehe auch Cioffi-Revilla 2010, S. 259). Doch auch hier wird schnell klar, dass es sich nicht zwangsläufig um neue Entwicklungen handelt. Im Gegenteil: Es lässt sich mittlerweile bereits eine historische Perspektive auf diesen Forschungsbereich einnehmen (Cioffi-Revilla 2017, S. 18 ff.). Bereits in den 1960er-Jahren finden sich Publikationen, in denen diskutiert wird, inwiefern computerbasierte Methoden für sozialwissenschaftliche Fragestellungen nutzbar sind (Coleman 1964).

Im Zusammenhang mit Computational Methods haben sich in verschiedenen Disziplinen Bezeichnungen herausgebildet, mit denen die Verbindung zu compu-

terbasierten Methoden angezeigt wird (Tab. 1.1; siehe auch Welker 2019). Ganz allgemein wird in den Sozialwissenschaften von Computational Social Science (CSS) und in den Geisteswissenschaften von Digital Humanities (DH) gesprochen[2]. Daneben finden sich fachspezifische Bezeichnungen wie Digital Sociology, Computational Communication Science, Digital History oder Computational Linguistics. Die englischen Bezeichnungen machen deutlich, dass es sich um globale Entwicklungen handelt.

Tab. 1.1 Forschungsrichtungen, in denen Computational Methods eingesetzt werden

Begriff	**Definition (Beispiel)**
Computational Social Science (CSS)	„The new field of Computational Social Science can be defined as the interdisciplinary investigation of the social universe on many scales, ranging from individual actors to the largest groupings, through the medium of computation“ (Cioffi-Revilla 2017, S. 2).
Digital Sociology	„The ‚digital‘ in digital sociology may denote at least three different things: it may refer to (1) the *topics* of social enquiry; (2) the *instruments and methods* of social research; (3) the *platforms* for engaging with the audiences and publics of sociology. Depending on which of these aspects of the ‚digital‘ we consider, we arrive at a very different understanding of what digital sociology is“ (Marres 2017, S. 24).
Computational Communication Science (CCS)	„Computational communication science studies generally involve: (1) large and complex data sets; (2) consisting of digital traces and other ‚naturally occurring‘ data; (3) requiring algorithmic solutions to analyze; and (4) allowing the study of human communication by applying and testing communication theory“ (van Atteveldt und Peng 2018, S. 82; in Anlehnung an Shah et al. 2015).
Digital Humanities (DH)	„‚Digital Humanities‘ covers a great number of activities in research, teaching, and cultural production, using computers and web-based platforms as new technologies for both scholars and the general public. New modes of digital collaboration and sharing are re-defining the lines between academic and cultural production“ (DHI 2020). „[D]ie Summe aller Versuche, die Informationstechniken auf den Gegenstandsbereich der Geisteswissenschaften anzuwenden“ (Jannidis et al. 2017, S. 13).

(Fortsetzung)

[2] Zur Bedeutung des Begriffs Digital Humanities gibt es eine umfangreiche fachliche Debatte, siehe die Beiträge in Terras et al. (2013).

Tab. 1.1 (Fortsetzung)

Begriff	**Definition (Beispiel)**
Humanities Computing	„Humanities computing is precisely the automation of every possible analysis of human expression (therefore, it is exquisitely a ‚humanistic' activity), in the widest sense of the word, from music to the theater, from design and painting to phonetics, but whose nucleus remains the discourse of written texts" (Busa 2004, S. xvi).
Digital History	„[D]igital history is the process by which historians are able to use computers to do history in ways impossible without the computer" (Burton 2005, S. 207). „The most important weapon for building the digital future we want is to take an active hand in creating digital history in the present" (Cohen und Rosenzweig 2006).
Computational Linguistics	„Computational linguistics is the scientific and engineering discipline concerned with understanding written and spoken language from a computational perspective, and building artifacts that usefully process and produce language, either in bulk or in a dialogue setting" (Schubert 2020).
Digital Methods	„[T]he repurposing of methods in media for social and cultural research" (Rogers 2010, S. 243).
Webometrics	„Webometrics […] covers research of all network-based communication using informetric or other quantitative measures" (Almind und Ingwersen 1997, S. 404). „The study of the quantitative aspects of the construction and use of information resources, structures and technologies on the Web drawing on bibliometric and informetric approaches" (Björneborn 2004, S. 12; siehe auch Björneborn und Ingwersen 2004). „[T]he study of web-based content with primarily quantitative methods for social science research goals using techniques that are not specific to one field of study" (Thelwall 2009, S. 6).
Data Mining	„Data mining is the application of specific algorithms for extracting patterns from data" (Fayyad et al. 1996, S. 39).
Knowledge Discovery in Databases (KDD)	„KDD is the nontrivial process of identifying valid, novel, potentially useful, and ultimately understandable patterns in data" (Fayyad et al. 1996, S. 40).
Data Science	„[D]ata science is a new interdisciplinary field that synthesizes and builds on statistics, informatics, computing, communication, management, and sociology to study data and its environments (including domains and other contextual aspects, such as organizational and social aspects) in order to transform data to insights and decisions by following a data-to-knowledge-to-wisdom thinking and methodology" (Cao 2017, S. 8).

Quelle: eigene Darstellung

Weitere Bezeichnungen stammen stärker aus der Tradition der Informatik und fokussieren spezifische Aspekte der Datenerhebung, wie etwa die Begriffe Data Mining oder Knowledge Discovery in Databases. Diese Begriffe findet man auch außerhalb der und an der Schnittstelle zur Wissenschaft. Als Bezeichnung für in der Wirtschaft verbreitete Tätigkeitsfelder haben sich die Begriffe Data Engineer und Data Scientist herausgebildet, inklusive entsprechender Studiengänge. Während erstere vor allem mit der Entwicklung von Software für die Datenaufbereitung beschäftigt sind, widmen sich letztere der Analyse. Hinzu kommen in größeren Organisationen Data Architects, die vor allem die Infrastruktur und Koordination der Teilaufgaben im Blick haben.

Dass sich in diesem Bereich verschiedene Traditionen verbinden, erkennt man auch daran, dass je nach Perspektive unterschiedliche Terminologien für die gleichen Sachverhalte verwendet werden.[3] Wo in der klassischen Statistik von unabhängigen und abhängigen Variablen die Rede ist, sprechen Data Scientists von Features und Labels. Beide Varianten umfassen die Merkmalsausprägungen der untersuchten Fälle, etwa das Erstelldatum von Kommentaren auf einer Webseite. Dabei geht es jedoch um mehr als nur um alternative Bezeichnungen, häufig unterscheiden sich auch die Zielstellungen. Während aus anwendungsorientierter Sicht vor allem die optimale Vorhersage von Verhalten relevant ist, suchen Wissenschaftler:innen nach Erklärungen. Beide Perspektiven modellieren die Wirklichkeit mit statistischen Mitteln. Erstere streben dabei allerdings eine möglichst hohe Modellgüte an, während für letztere die einzelnen Parameter der Modelle wichtiger sind.[4] Auch innerhalb der Wissenschaft kommen verschiedene erkenntnistheoretische Grundpositionen zusammen. Der Begriff Digital Methods kommt beispielsweise aus einer stärker interpretativen oder sogar ethnografischen Perspektive, wenn darunter „the repurposing of methods in media for social and cultural research" (Rogers 2010, S. 243) verstanden wird. Damit ist etwa gemeint, dass für die wissenschaftliche Analyse von Online-Kommunikation die Mittel der Online-Kommunikation verwendet werden sollten. So würde man Suchmaschinen untersuchen, indem man Suchmaschinen verwendet. Hinzu kommt vor allem im Bereich der Digital Humanities noch eine ganz praktische Perspektive, wenn darunter auch künstlerisches oder literarisches

[3] Zur unterschiedlichen Terminologie siehe die Einführung von Attewell und Monaghan (2015, S. 8 ff.).

[4] Die unterschiedlichen Perspektiven lassen sich gut am Umgang mit Regressionsmodellen verdeutlichen. Während Data Scientists den Determinationskoeffizienten R^2 optimieren und die ohnehin unübersichtliche Anzahl an Variablen bzw. Einflussfaktoren hintenanstellen, sind Sozial- und Geisteswissenschaftler:innen vorrangig an den Beta-Koeffizienten der einzelnen Variablen interessiert und akzeptieren dabei eine geringe Erklärungskraft des Gesamtmodells.

Schaffen verstanden wird, bei dem computergestützt kulturelle Artefakte nicht nur beschrieben oder analysiert, sondern selbst produziert werden.

Die Kombinationen mit dem Wort „digital“ verweisen bereits auf den Gegenstandsbereich digitaler Kommunikation, der allerdings sehr umfassend und damit schwer fassbar ist.[5] Etwas engere Fokussierungen nehmen die Begriffe Internet Research oder noch enger Web Science und Web Mining vor.[6] Noch spezifischer sind Felder wie Webometrics, in denen in der Tradition bibliografischer Zitationsanalysen unter anderem netzwerkanalytische Strukturen zwischen Webseiten im Vordergrund stehen. In der vorliegenden Einführung wird der nicht auf eine Disziplin festgelegte Begriff Computational Methods verwendet, da die Methoden im Vordergrund stehen sollen. Diese Methoden können in kreativer Weise für sehr unterschiedliche Fragestellungen fruchtbar gemacht werden.

Wenn auch die inhaltliche Begriffsbestimmung nicht trivial ist, so sind Computational Methods faszinierend. Ein Teil der Faszination ergibt sich vermutlich daraus, dass Computer (als technische Maschinen) dabei Aufgaben übernehmen, die im ersten Moment menschliche Fähigkeiten voraussetzen. Deutlich wird dies im Bereich der Sprach- und Bilderkennung: Suchmaschinen können Bilder danach sortieren, ob sie Katzen oder Menschen enthalten. Smartphones reagieren auf die verbale Anweisung, einen Timer zu stellen. Und in menschlichen Gesichtern wird versucht, automatisch Emotionen zu erkennen. In diesen Beispielen werden maschinell komplexe, aber sehr spezifische Aufgaben gelöst, die man herkömmlich nur Menschen zutraut. Deshalb wird auch von künstlicher Intelligenz gesprochen, ohne dass dabei tatsächlich Intelligenz im Spiel wäre (Russell und Norvig 2012, S. 22).

Interessant für die wissenschaftliche Analyse werden entsprechende Verfahren unter anderem dadurch, dass sie automatisiert auf einer großen Datenmenge durchgeführt werden können. Hierin liegt ein entscheidender Vorteil gegenüber menschlichen Analysen, wenn eine kleine Stichprobe für eine Fragestellung nicht ausreicht. Zudem versprechen Computational Methods eine hohe Reliabilität – das heißt, sie führen zumindest dem ersten Anschein nach zu den immer gleichen Ergebnissen.[7]

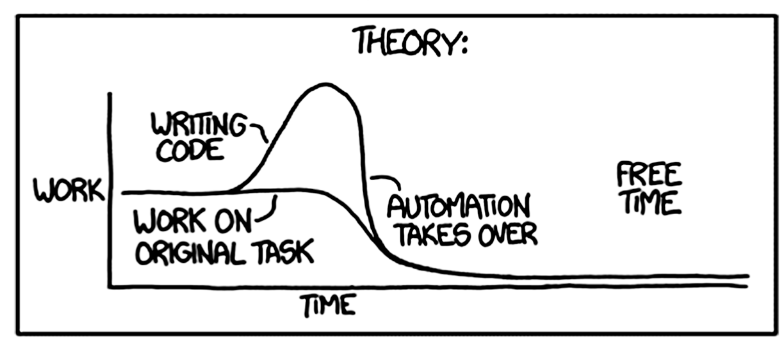
---

[5] Für kritische Anmerkungen zur Verwendung des Digitalisierungsbegriffs siehe Jünger und Schade (2018).

[6] Das Internet und das Web unterscheiden sich sowohl in technischer als auch in organisatorischer Hinsicht (Beck 2006, S. 30), die Begriffe sollten deshalb nicht verwechselt werden.

[7] Tatsächlich ist Reliabilität im Sinne von Reproduzierbarkeit häufig nicht gegeben, da sich die untersuchten Gegenstände und die Methoden schnell wandeln (Jünger 2018, S. 122). Selbst wenn im Sinne von Open Science die Quelltexte von Programmen veröffentlicht werden, lassen sich Skripte nach einigen Jahren aufgrund veränderter technischer (z. B. Betriebssysteme) und organisatorischer (z. B. Plattformen) Rahmenbedingungen häufig nicht mehr ausführen.

Theoretisch besteht die Besonderheit von Automatisierung darin, dass mit einem vergleichsweise hohen Anfangsaufwand ein System aufgesetzt wird, das anschließend ohne menschliche Eingriffe läuft und so den Aufwand reduziert. Automatisierung bedeutet zum Beispiel bei der Analyse von Texten, dass der Zusatzaufwand mit jedem zusätzlichen Dokument sinkt. Aus ökonomischer Sicht sinken die Grenzkosten: „As a rule-of-thumb, we consider a system fully automated if the marginal cost of analyzing additional texts goes to zero as the size of the corpus being analyzed increases, and the coding is completely replicable given a set of software, dictionaries, and so forth“ (Monroe und Schrodt 2008, S. 352; siehe auch Scharkow 2011, S. 547). Automatisierung ist allerdings in der Praxis nicht zwangsläufig effektiv oder effizient (Abb. 1.1). Das Lernen der Verfahren, die



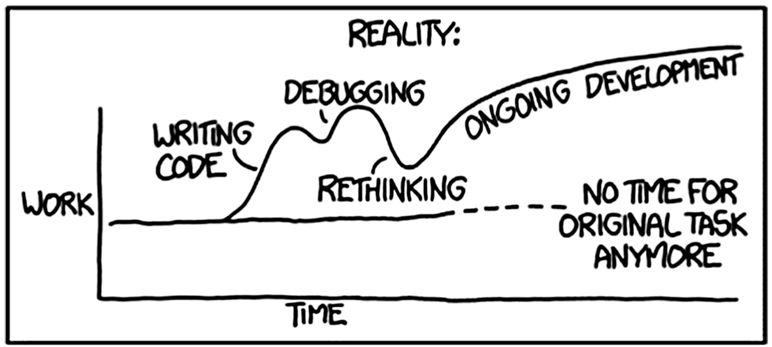


Abb. 1.1 Der Fluch der Automatisierung. (Quelle: Munroe (2013; https://xkcd.com/1319/))

Vorbereitung der Infrastruktur und die Behebung von Fehlern führen häufig dazu, dass kontinuierlicher Aufwand betrieben werden muss. Gleichzeitig verändern sich dabei die Fragestellungen und Probleme, an denen gearbeitet wird. Insofern ist die Beschäftigung mit Computational Methods trotz oder vielleicht gerade aufgrund von Bemühungen zur maschinellen Automatisierung ein kreativer und inspirierender Prozess.

Automatisierung kann in allen Phasen des Forschungsprozesses eine Rolle spielen:

- Mittels automatisierter **Datenerhebung** können Inhalte wie etwa Kommentare auf Social-Media-Plattformen oder Webseiten in großem Umfang erschlossen werden. Zu diesem Zweck wird Webscraping eingesetzt, bei dem im Prinzip der Browser so automatisiert wird, dass die Klicks eines Menschen simuliert werden. Aus den Quelltexten der angesurften Webseiten werden dann die gewünschten Daten extrahiert. Einige Anbieter stellen auch Application Programming Interfaces (APIs) bereit, über die vorstrukturierte Daten abgerufen werden können.
- Automatisierte **Datenaufbereitung** meint die Umwandlung unstrukturierter Inhalte in strukturierte Daten. Unstrukturierte Inhalte zeichnen sich dadurch aus, dass die Eigenschaften der Fälle nicht bereits in standardisierter Form vorliegen. Bei der automatisierten Textanalyse werden beispielsweise aus Kommentaren die Sätze und Wörter extrahiert und in Datensätze umgeformt, sodass Wörter zu Variablen werden (Document-Term-Matrix). Bei der Datenaufbereitung geht es auch darum, nicht benötigte Daten zu entfernen (engl. *boilerplate removal*).
- Der Übergang zwischen automatisierter Aufbereitung und **Datenanalyse** ist fließend. Auf Grundlage strukturierter Textdaten lassen sich beispielsweise Kommentare danach klassifizieren, ob sie eher positive oder eher negative Aussagen enthalten. Hier unterscheidet man überwachte und unüberwachte Lernverfahren. Erstere zeichnen sich dadurch aus, dass die Zielkategorien durch menschlich erstelltes Trainingsmaterial vorgegeben werden. Letztere sind explorativ angelegt und gruppieren Fälle nach Ähnlichkeit, um zum Beispiel Themen zu bestimmen. In das Feld automatisierter Datenanalyse fallen darüber hinaus Netzwerkanalysen, Zeitreihenanalysen, geografische Analysen und Computersimulationen.[8]

[8] Für den Bereich Computational Social Science unterscheidet Cioffi-Revilla (2010, S. 260) automatische Informationsextraktion, Netzwerkanalysen, geografische Informationssysteme, Modellierung von Komplexität und soziale Simulationsmodelle.

- Bei der **Darstellung** von Ergebnissen können ebenfalls computerbasierte Methoden zum Einsatz kommen. Beispielsweise werden große Netzwerke häufig mit Algorithmen visualisiert, bei denen die Elemente schrittweise in eine zweidimensionale Anordnung gebracht werden, sodass verbundene Elemente möglichst nah beieinanderstehen. Darüber hinaus lassen sich Ergebnisse interaktiv aufbereiten und online veröffentlichen, damit einzelne Datenkategorien oder Parameter von den Nutzer:innen nachträglich angepasst werden können.

In diesem Sinne wird der Begriff im vorliegenden Buch verwendet: Der Bereich Computational Methods umfasst alle Verfahren der *automatisierten* Datenerhebung, -aufbereitung, -analyse und -darstellung.[9] Im Buch werden einerseits konzeptionelle Grundlagen vermittelt und andererseits praktische Anleitungen gegeben. Eine lineare Lektüre ist dabei nicht unbedingt nötig. Sie können beispielsweise direkt in die Kapitel zum Webscraping einsteigen. Je nach Vorwissen wird es hilfreich sein, von dort gezielt in die vorangegangenen Kapitel zu springen, vor allem, wenn Ihnen die verwendeten Begriffe, Datenformate und Programmiertechniken unbekannt sind. In Kap. 12 am Ende des Buchs finden Sie weitere Hinweise zu möglichen Leserichtungen.

1.2 Der Werkzeugkoffer

Zwei Werkzeuge sind nicht nur für die folgenden Kapitel, sondern auch sonst für die Arbeit mit Computational Methods unverzichtbar: die Kommandozeile und ein guter Texteditor. Beides wird im Folgenden kurz eingeführt. Sie können die Ausführungen gegebenenfalls zunächst überfliegen und später nachschlagen. Zumindest die im ersten Abschnitt genannten Grundregeln für den Umgang mit Dateien sollten Sie aber möglichst frühzeitig umsetzen. Im letzten Abschnitt zu den Begleitmaterialien finden Sie außerdem Hinweise zu Daten und Skripten, die ergänzend zu den einzelnen Kapiteln als Hilfsmittel zur Verfügung stehen.

Dieses Buch führt damit in die Tiefe der Computational Methods ein und soll dazu beitragen, dass Sie entsprechende Werkzeuge irgendwann auch selbst entwickeln können. Alternativ finden sich mittlerweile unter dem Stichwort Forschungssoftware (engl. *research software*) viele leistungsfähige Tools mit grafischen Be-

[9] Die Division der ICA definiert ähnlich: „Computational methods cover computerized tools and algorithms for collecting, processing, analyzing, and visualizing data such as social media data, news sites, and other forms of communication“ (ICA CM 2017).

nutzeroberflächen.[10] Diese sind sehr hilfreich für einen schnellen Einstieg und als Inspiration für mögliche Analyseverfahren. Wenn Sie sich in einen Bereich neu einarbeiten – beispielsweise in die Netzwerkanalyse –, dann lohnt es sich, zunächst nach passender Software zu recherchieren. Für viele Bereiche haben andere Wissenschaftler:innen bereits Materiallisten zusammengestellt, die häufig als Awesome List bezeichnet werden.[11] Allerdings sind die Möglichkeiten dann auf die vorgegebenen Funktionen begrenzt und auch nicht immer im Detail transparent. Diese Beschränkungen können Sie überwinden, indem Sie selbst die Grundtechniken von Computational Methods erlernen.

1.2.1 Grundregeln für Ordner und Dateien

Wichtig ist für die Arbeit mit Daten, dass Sie sich mit dem Dateisystem Ihres Computers auskennen. Einige Grundregeln erleichtern die Arbeit ungemein:

1. Erstellen Sie ein **Arbeitsverzeichnis**, das Sie leicht erreichen können, beispielsweise in dem Dokumente-Ordner Ihres Computers. Unter einem Verzeichnis versteht man einen Ordner, in dem Unterordner und Dateien abgelegt werden. Die Anzahl der Dateien wächst in Programmierprojekten schnell an. Nutzen Sie deshalb zur Organisation auch Unterverzeichnisse. Arbeiten Sie immer in diesem Verzeichnis, beispielsweise wenn Sie die Kommandozeile starten.

   Zu Ordnern, Unterordnern und Dateien können Sie mit der Maus navigieren, indem Sie im Explorer (Windows) bzw. dem Finder (Mac)[12] die entsprechenden Symbole anklicken. Es ist allerdings auch möglich, Dateien oder Verzeichnisse gezielt über den sogenannten Pfad zu adressieren. Der Aufbau von Pfaden unterscheidet sich zwischen den Betriebssystemen. Unter Windows (Abb. 1.2) beginnt ein Pfad mit dem Buchstaben des Laufwerks gefolgt von einem Doppelpunkt (meistens *C:*). Danach folgen getrennt mit Backslashs[13] die Unterord-

[10] Siehe zum Beispiel die Sammlungen von Social Media Data Stewardship (2021; https://socialmediadata.org/social-media-research-toolkit/) der Deutsche Gesellschaft für Publizistik- und Kommunikationswissenschaft (2022; https://www.dgpuk.de/de/forschungssoftware.html). Meine Forschungssoftware. Zugriff am 16.05.2022.

[11] Zum Beispiel: Briatte (2021; https://github.com/briatte/awesome-network-analysis).

[12] Der Explorer bzw. Finder sind die Programme, mit denen man auf die Laufwerke und die eigenen Dateien zugreift.

[13] Unter Windows werden Pfadelemente mit dem Backslash \ getrennt, unter Linux und Mac dagegen mit einfachem Slash / . Der Vorwärtsslash / funktioniert in der Regel aber auch unter Windows.

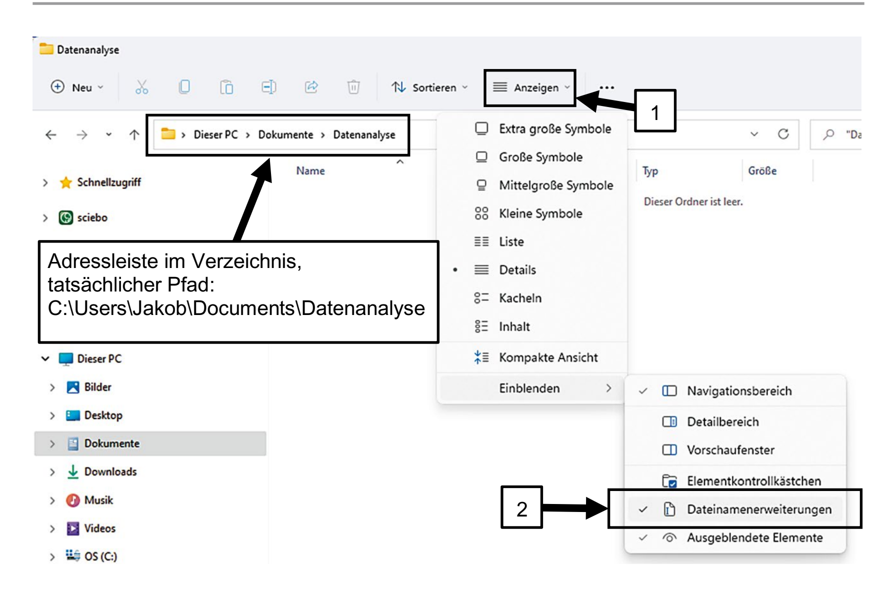


Abb. 1.2 Die Adressleiste und das Einblenden von Dateierweiterungen unter Windows 11. Hinweis: Beachten Sie den Unterschied zwischen dem angezeigten Pfad in der Adressleiste und dem tatsächlichen Pfad. (Quelle: eigene Darstellung)

ner. Der Pfad ist im Explorer in der Adressleiste erkennbar. Zu beachten ist, dass die Anzeige häufig gekürzt ist und englische Ordnernamen übersetzt sind. Der tatsächliche Pfad kann herausgefunden werden, indem man mit der Maus innerhalb eines Unterordners in die Adressleiste klickt. Unter Mac und Linux beginnt ein Pfad mit einem Slash, danach folgen getrennt mit weiteren Slashs die Unterordner. Eine vollständige Angabe ist hier aber nur selten nötig, denn mit einer Tilde ~ kann direkt auf das aktuelle Benutzerverzeichnis verwiesen werden (Abb. 1.3).

Wenn Sie bereits im gewünschten Verzeichnis sind, können Sie in der Kommandozeile oder beim Programmieren relative Pfade einsetzen. Relative Pfade beginnen im aktuellen Verzeichnis. Aus dem aktuellen Arbeitsverzeichnis und dem relativen Pfad wird automatisch der absolute Pfad zusammengesetzt. Wenn beispielsweise der Ordner *C:\Users\Jakob\Documents* Ihr aktuelles Arbeitsverzeichnis ist, dann reicht die Angabe *Datenanalyse*, um das entsprechende Unterverzeichnis zu benennen.

2. Verwenden Sie in **Dateinamen** keine Leerzeichen, Umlaute und Sonderzeichen. Anstelle von Leerzeichen können Sie Unterstriche verwenden. Unter Mac

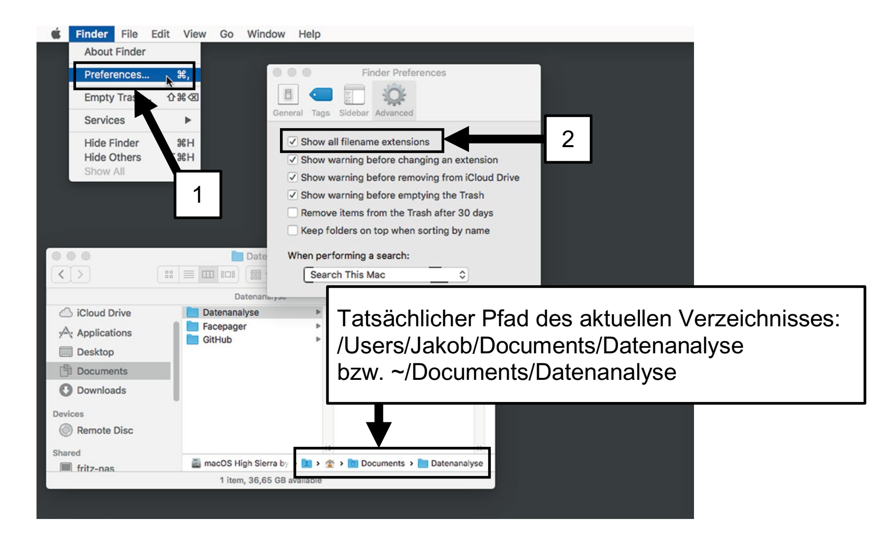


Abb. 1.3 Dateierweiterungen unter macOS im Finder einblenden. Hinweis: Beachten Sie, dass sich die Tilde im tatsächlichen Pfad auf das Benutzerverzeichnis bezieht. (Quelle: eigene Darstellung)

und Linux wird Groß- und Kleinschreibung unterschieden. Deshalb ist es günstig, alle Namen in Kleinschreibung zu halten.

Dateien setzen sich aus einem Namen und einer Endung zusammen. Die Dateiendung umfasst den letzten Teil nach dem Punkt; darüber ist der Dateityp erkennbar. Bilder haben beispielsweise die Endung *.png* oder *.jpg*. Quelltexte enden mit *.R* oder mit *.py*. Daten werden häufig in Dateien mit den Endungen *.csv* und *.json* abgelegt. Textdateien erkennen Sie an der Endung *.txt*. Auch Markdown-Dateien mit der Endung *.md* werden Ihnen begegnen. Diese enthalten Text, der mit einfachen Konventionen so strukturiert und formatiert ist, dass er zum Beispiel auf Webseiten angezeigt werden kann (siehe Kap. 3).

Je nach Voreinstellung Ihres Betriebssystems müssen Sie die Dateiendungen erst einblenden, das sollten Sie am besten jetzt sofort tun! Unter Windows 11 (Abb. 1.2) öffnen Sie dazu den Explorer, wechseln in den Reiter ANZEIGEN und wählen im Bereich EINBLENDEN die Option DATEINAMENERWEITERUNGEN. Unter macOS (Abb. 1.3) öffnen Sie im Finder die Voreinstellungen und wählen die Option ALLE DATEINAMENSUFFIXE EINBLENDEN. Unter Linux-Systemen werden die Erweiterungen normalerweise standardmäßig angezeigt.

1.2.2 Die Kommandozeile

Mit der Kommandozeile, auch Eingabeaufforderung, Konsole, Terminal oder Shell genannt, schauen Sie hinter den Vorhang der grafischen Oberfläche von Windows, Mac oder Linux. Hier werden über die Tastatur Befehle eingegeben, mit denen zum Beispiel Programme installiert oder gestartet werden. Die Handhabung und die Befehle unterscheiden sich etwas zwischen den Betriebssystemen. Auch wenn Sie normalerweise unter Windows oder Mac arbeiten, kommt vermutlich irgendwann der Zeitpunkt, zu dem Sie in ein Linux-Terminal wechseln müssen. Denn insbesondere Cloud-Computing, das heißt die Nutzung von Servern statt des eigenen Computers, setzt üblicherweise auf einer Infrastruktur mit Linux-Systemen auf.

Die klassische Kommandozeile[14] öffnen Sie unter **Windows** zum Beispiel, indem Sie die Windows-Taste drücken und die Buchstaben `cmd` (= command) eingeben. Nach dem Starten wird das aktuelle Verzeichnis angezeigt (Abb. 1.4). Hinter dem aktuellen Verzeichnis blinkt ein Cursor und Sie können an dieser Stelle Be-

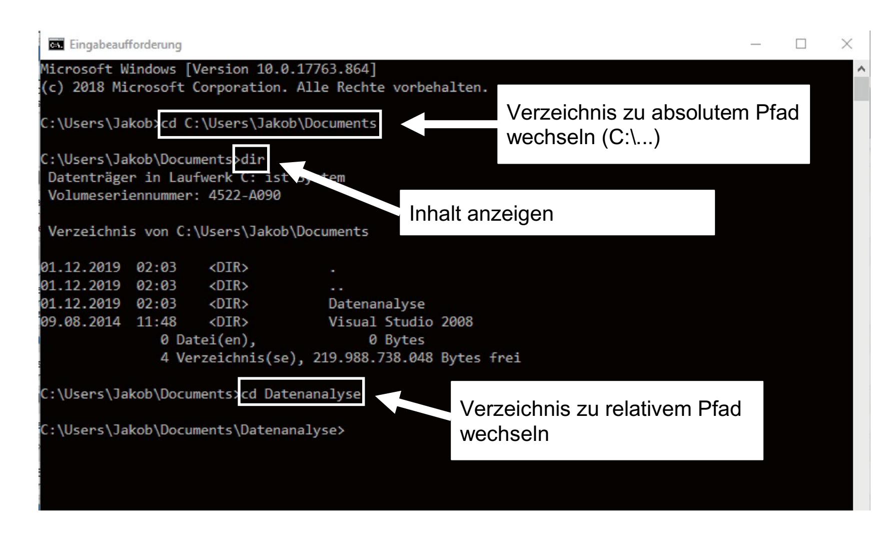


Abb. 1.4 Die Eingabeaufforderung unter Windows 10. (Quelle: eigene Darstellung)

---

[14] Unter Windows steht außerdem die mächtigere PowerShell zur Verfügung, die Befehle unterscheiden sich teilweise von der Eingabeaufforderung. Darüber hinaus können Shells installiert werden, die der Linux- und Mac-Shell gleichen. Wenn Sie die Versionsverwaltung Git (siehe unten) installieren, steht anschließend eine Linux-ähnliche Shell zur Verfügung.

fehle eingeben, um beispielsweise Verzeichnisse zu wechseln oder Programme zu installieren. Ein Befehl wird ausgeführt, sobald er mit der Enter-Taste bestätigt wird.

Das Verzeichnis wechseln Sie mit dem Befehl `cd` (= change directory) und der Angabe eines absoluten oder relativen Pfads.[15] Der Befehl `dir` zeigt den Inhalt des Verzeichnisses an, also die enthaltenen Dateien und Ordner. Um vom aktuellen Verzeichnis eine Ebene nach oben zu wechseln, wird der Befehl `cd ..` verwendet. Längere Pfade oder Befehle können auch über die Zwischenablage in die Kommandozeile eingefügt werden.

Den Umweg über den `cd`-Befehl kann man sich aber sparen, wenn man das Verzeichnis zunächst im Explorer öffnet. Gibt man anschließend oben in die Adressleiste `cmd` ein und bestätigt mit der Entertaste, dann wird die Kommandozeile in diesem Verzeichnis geöffnet.

Auf dem **Mac** lässt sich die Kommandozeile öffnen, indem in der Spotlight-Suche (Lupen-Symbol) „Terminal" eingegeben wird. Im Terminal wird zu Beginn einer Zeile der Name des Computers angezeigt, nach einem Doppelpunkt folgen der Name des aktuellen Verzeichnisses, dann ein Leerzeichnen und der Benutzername (Abb. 1.5). Die Befehle werden hinter dem Dollarzeichen eingegeben. Auch hier werden Verzeichnisse durch den Befehl `cd` gefolgt von einem relativen oder absoluten Pfad gewechselt. Der Inhalt des aktuellen Verzeichnisses wird mit dem Befehl `ls` angezeigt, wobei der Parameter `-l` für eine kompakte Darstellung sorgt: `ls -l`.

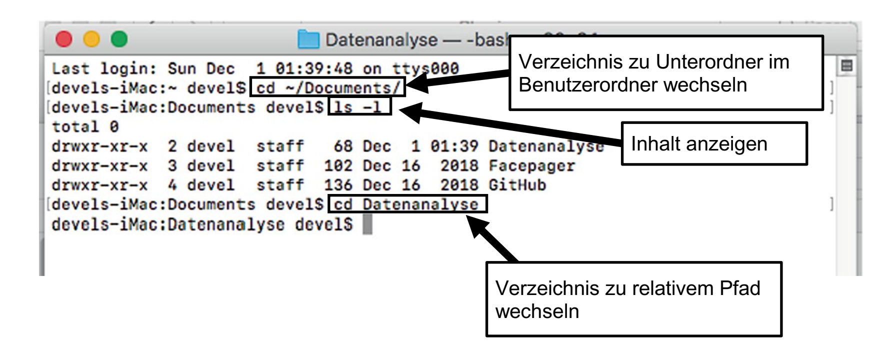


Abb. 1.5 Das Terminal unter macOS High Sierra. (Quelle: eigene Darstellung)

[15] Das Laufwerk können Sie nicht über den Befehl `cd` wechseln, stattdessen muss der Buchstabe mit Doppelpunk ohne weiteren Befehl eingegeben werden.

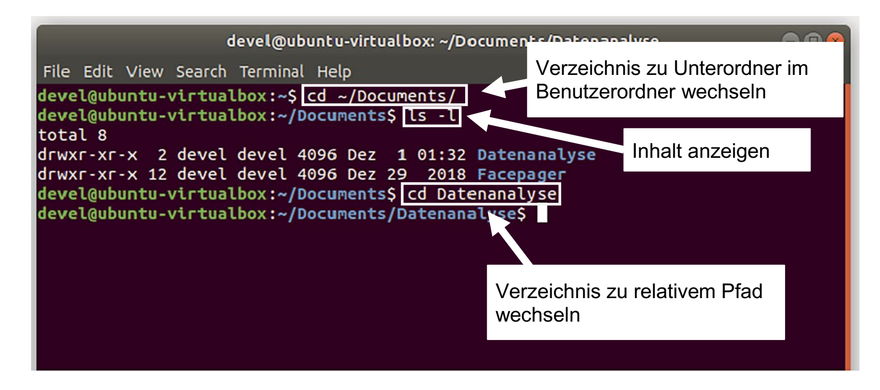


Abb. 1.6 Das Terminal unter Ubuntu 18. (Quelle: eigene Darstellung)

Unter einem **Linux**-System wie Ubuntu erreichen Sie die Kommandozeile über die Command-Taste und Eingabe von „Terminal". Da sowohl macOS als auch Linux zu den Unix-Systemen zählen, ist die Bedienung identisch (Abb. 1.6).

Egal auf welchem System Sie arbeiten, einige Funktionen sind in nahezu jeder Kommandozeile enthalten. Mit den Pfeiltasten (hoch/runter) können Sie etwa vorherige Befehle aufrufen. Bei langen Verzeichnisnamen hilft auch die Autovervollständigung: Tippen Sie die ersten Buchstaben ein und drücken Sie dann die Tabulatortaste!

Die Kommandozeile können Sie schließen, indem Sie das Fenster schließen. Wenn Sie innerhalb der Kommandozeile in einem Befehl festhängen, dann können Sie in der Regel über die Tastenkombination Strg + Pause oder Strg + C (Windows) bzw. Command + Pause oder Command + C (Mac) entkommen.[16]

1.2.3 Texteditoren

Ein weiteres wichtiges Werkzeug sind Texteditoren. Denn viele Datenformate, Skripte und Quelltexte sind Textformate. Vor allem wenn man den Inhalt einer Datei oder das Dateiformat nicht kennt, sollte man die Datei zunächst mit einem Texteditor erkunden. Auf der Kommandozeile stehen unter Unix-Systemen häufig die Editoren `vim` und `nano` zur Verfügung.

[16] Das Pluszeichen bei der Angabe von Tastenkombinationen bedeutet, dass die ersten Tasten gehalten werden, es wird nicht mit eingegeben.

Für Einsteiger ist die Arbeit auf der Kommandozeile aber häufig etwas umständlich. Ein für alle Betriebssysteme geeigneter Open-Source-Texteditor ist Atom.[17] Nach dem ersten Starten wird eine Einführung präsentiert. Eine Stärke dieses Editors ist die Paketverwaltung (Menüpunkt EINSTELLUNGEN), über die viele Erweiterungen nachinstalliert werden können. Unter Windows ist Notepad++ empfehlenswert und die meisten Beispiele in diesem Buch werden mit diesem Editor illustriert.[18] Ein Texteditor, der speziell für MacOS entwickelt wurde, ist Textmate.[19] Eine weitere betriebssystemübergreifende Alternative bietet der Editor VS Code.[20] Installieren Sie sich am besten jetzt gleich einen solchen Texteditor!

Um Dateien in einem Texteditor zu öffnen, gibt es in der Regel zwei Wege. Entweder starten Sie zuerst den Editor und öffnen die Datei über das entsprechende Menü. Oder Sie suchen im Dateimanager (Explorer, Finder bzw. Files) das Verzeichnis und öffnen die Datei von dort aus über das Kontextmenü der Datei. Dieses Kontextmenü wird in der Regel über die rechte Maustaste erreicht. Dort finden Sie beispielsweise unter Windows einen Punkt ÖFFNEN MIT und können dann den Texteditor auswählen. Erscheint der Texteditor nicht bereits bei den vorgeschlagenen Programmen, können Sie ihn dort hinzufügen. Dafür suchen Sie den Ordner, in welchem der Editor installiert ist und wählen in diesem unter Windows die Datei mit der Endung *.exe* aus.

1.2.4 Begleitmaterialien zum Buch

Ein weiteres Hilfsmittel für den Einstieg in die Welt der Computational Methods können die Begleitmaterialien zu diesem Lehrbuch sein. Da Computational Methods sehr praktisch sind, sollen die vorbereiteten Beispiele, Skripte und Datensätze die einzelnen Kapitel dieses Buchs ergänzen und dabei helfen, schrittweise eigene praktische Kompetenzen aufzubauen. Besonders bei komplexen Verfahren kann es hilfreich sein, zunächst vorbereitete Skripte Schritt für Schritt nachzuvollziehen, bevor man selbst Anpassungen vornimmt und schlussendlich eigene Skripte schreibt.

Die Begleitmaterialien befinden sich in einem GitHub-Repositorium,[21] das zu Beginn eines jeden Kapitels verlinkt ist und auf das wir im Text mit ☛ *Reposito-*

---

[17] Siehe GitHub (2022a; https://atom.io/).

[18] Siehe Ho (2022; https://notepad-plus-plus.org/).

[19] Siehe MacroMates (2021; https://macromates.com/).

[20] Siehe Microsoft (2022b; https://code.visualstudio.com/).

[21] GitHub ist eine Plattform für Entwickler:innen, die dort ihren Code teilen, die Entwicklungsarbeit dokumentieren und koordinieren (siehe Abschn. 6.1).

rium verweisen. Die Inhalte des Repositoriums können Sie entweder im Browser öffnen oder lokal auf Ihrem Computer speichern und bearbeiten. Um die Dateien gesammelt herunterzuladen, finden Sie unter der Bezeichnung CODE einen Link zu einer Zip-Datei. Da Repositorien auf GitHub unter Versionsverwaltung stehen, können diese auch über Befehle der Versionsverwaltung heruntergeladen werden. Nutzen Sie die Gelegenheit, um das Zusammenspiel von Kommandozeile, Versionsverwaltung und Texteditor auszuprobieren! Dazu installieren Sie zunächst die für Ihr Betriebssystem passende Git-Version.[22] Anschließend können Sie über die folgenden Schritte das Verzeichnis herunterladen:

1. Legen Sie ein Arbeitsverzeichnis auf Ihrem Computer an und öffnen Sie dort die Kommandozeile.
2. Laden Sie das Repositorium über die Kommandozeile mit folgendem Befehl herunter: `git clone https://github.com/strohne/cm`
3. Öffnen Sie in einem der Verzeichnisse des heruntergeladenen Repositoriums die Datei *readme.md* mit Notepad++, Atom oder einem anderen Texteditor.

Einen Einstieg in Versionsverwaltungen bietet Abschn. 6.1.

1.3 Mit Fehlern umgehen

Fehler zu machen, zu erkennen und zu beheben ist ein ganz wesentlicher Bestandteil von Computational Methods. Schließlich handelt es sich um ein interdisziplinäres Feld, das Wissen und Methoden aus unterschiedlichen Bereichen umfasst, in denen man nicht zwangsweise bereits Expert:in ist. Außerdem lassen sich kaum neue Daten erschließen, Programme erkunden oder Methoden ausprobieren, ohne dabei auch einmal festzustecken oder zeitweise in die falsche Richtung zu laufen.

Während das Lösen von Problemen durchaus viel Spaß bereiten kann, wenn man an neuen Ideen knobelt und dabei über sich hinauswächst, kann es gleichzeitig frustrierend und zeitintensiv sein. Um möglicher Frustration vorzubeugen, sind nachfolgend einige häufige Fehler und Tipps aufgeführt:

- Manchmal werden Dateien oder Verzeichnisse nicht gefunden. Hier hilft es, systematisch zu überprüfen, ob sich der angegebene Pfad auch auf das richtige Arbeitsverzeichnis bezieht, die Datei auch wirklich in diesem Ordner liegt und das Datenformat korrekt ist.

[22] Siehe Git (2022a; https://git-scm.com/downloads).

- Häufige Fehlerquellen sind außerdem Schreibfehler. Diese sind leicht zu übersehen, da sie in einigen Programmen nicht optisch hervorgehoben werden. Auch Leerzeichen, Groß- und Kleinschreibung machen in den meisten Programmiersprachen einen Unterschied. Es hilft oft, mehrfach zu prüfen, ob Verzeichnisse, Variablen oder Befehle richtig geschrieben sind. Einige Wörter wie „for" oder „in" sind in Programmiersprachen mit Funktionen belegt und sollten nur dafür verwendet werden.
- Wenn Dateien nicht korrekt eingelesen werden, kann dies an Steuerzeichen oder nicht sichtbaren Zeichen liegen. In CSV-Dateien signalisieren zum Beispiel Zeilenumbrüche, wann eine neue Zeile in einer Tabelle beginnt und diese Markierungen können je nach Betriebssystem unterschiedlich formatiert sein. Hier hilft es, die Datei im Texteditor zu öffnen, nicht sichtbare Zeichen einzublenden und zum Beispiel Zeilenumbrüche auszutauschen (siehe Abschn. 3.1).
- Bei der Arbeit mit R oder Python verwendet man häufig Funktionen, also definierte Abläufe von Befehlen, die andere Entwickler:innen in sogenannten Packages bereitstellen. Packages werden meist laufend weiterentwickelt. Deshalb kommt es vor, dass Befehle nach einiger Zeit veralten und nicht mehr funktionieren. Hinweise dazu, wie Sie die Version der Packages überprüfen und aktualisieren, finden Sie in den entsprechenden Kapiteln (siehe Kap. 5). Mitunter werden Funktionen sogar abgeschafft (engl. *deprecated*), dann muss man sie durch Alternativen ersetzen.
- Über viele Fehlermeldungen haben sich meistens bereits andere geärgert. Deswegen hilft es häufig, angezeigte Fehlermeldungen in eine Suchmaschine einzugeben. Eine Plattform, auf der viele Problemlösungen dokumentiert sind und über die man Hilfe zu speziellen Programmierfragen bekommt, ist Stack Overflow.[23] Dahinter steht eine aktive Community, die sich gegenseitig bei Problemen rund um das Programmieren hilft.
- Eine gute Inspirationsquelle sind Cheatsheets, die im Internet zu allen möglichen Themen und Programmiersprachen oder -paketen zu finden sind. Auf nur ein bis zwei Seiten werden übersichtlich die entsprechenden Hilfsmittel zusammengefasst und es lässt sich schnell ein Überblick über wichtige Funktionen gewinnen.

Die aufgeführten Punkte erscheinen vielleicht im ersten Moment trivial. Dennoch tappen selbst ausgewiesene Expert:innen immer wieder in die gleichen Fallen. Die Fehlerbehebung bleibt stets eine der wichtigsten Tätigkeiten und es ist hilfreich, dafür nach und nach eigene Routinen auszubilden.

[23] Siehe Stack Overflow (2022; https://stackoverflow.com).

1.4 Überblick über das Buch

Computational Methods, wie sie hier verstanden werden, schlagen eine Brücke zwischen automatisierten Verfahren und inhaltlichen Fragestellungen. Fängt man an damit zu arbeiten, geht ein Großteil der Zeit in die Aufbereitung von Daten, die Erkundung von Methoden und natürlich in die Behebung von Fehlern. Allein die Auseinandersetzung mit den Methoden ist inspirierend für den Forschungsprozess, am Ende geht es aber darum, Phänomene in der geistigen, kulturellen und sozialen Welt zu verstehen und zu erklären. Mit Computational Methods werden zum einen altbekannte Fragestellungen adressiert, etwa wie sich Öffentlichkeit über Diskurse im Zeitverlauf entfaltet. Sie werfen neues Licht auf diese Phänomene, indem beispielsweise größere Textkorpora analysiert werden können oder Diskurse als Netzwerke von Akteuren und Texten verstanden werden. Zum anderen ergeben sich in unseren Lebenswelten allein durch das Aufkommen von Online-Plattformen und den Einsatz sogenannter Künstlicher Intelligenz – dieser schillernde Begriff meint nichts anderes als automatisierte Verfahren – auch neue Forschungsfragen. In Bezug auf Öffentlichkeit stellt sich beispielsweise die Frage, inwiefern Empfehlungssysteme zu einer Fragmentierung von Öffentlichkeit führen, wenn Menschen personalisierten Inhalten ausgesetzt werden. Und der Einsatz von Bots führt auch zu grundsätzlichen theoretischen Fragen, etwa zu der Frage, inwiefern Programme und Algorithmen als handelnde Akteure begriffen werden können und wem eigentlich die Verantwortung für automatisiertes Verhalten zugeschrieben wird.

Computational Methods sind somit in der Lebenswelt anzutreffen, gleichzeitig aber auch Analyseinstrumente für wissenschaftliche Fragestellungen. Die Welt dieser Methoden entwickelt sich ständig weiter. Damit Sie für unterschiedliche Szenarien gewappnet sind, werden im ersten Teil des Buchs konzeptionelle Grundlagen eingeführt. Unabhängig von konkreter Software oder bestimmten Tools werden Denkmuster vermittelt – insbesondere geht es darum, die Welt durch eine eckige Brille als Ansammlung von Matrizen und Tabellen zu betrachten. Denn dies ist eine grundlegende Datenstruktur, mit der Daten erfasst, transformiert und analysiert werden können.

- **Kap. 2** führt in Zugänge **zur automatisierten Datenerhebung** ein und benennt exemplarische Datenquellen: Unstrukturierte Inhalte können aus Webseiten ausgelesen werden, vorstrukturierte Daten werden über Application Programming Interfaces (APIs) bereitgestellt und schließlich findet sich mittlerweile eine Vielzahl von Datenbanken mit fertig aufbereiteten Datensätzen.
- In **Kap. 3** werden **Datenformate** vorgestellt, denen man in der Welt der Computational Methods begegnet. Grundlegend ist die Unterscheidung von Daten-

typen – beispielsweise von Zahlen oder Buchstaben. Diese Daten können tabellarisch zusammengestellt oder durch Auszeichnungssprachen und Objektdatenformate strukturiert sein. Auch für die Zusammenstellung mehrerer Tabellen zu Datenbanken gibt es etablierte Verfahren. Dahinter liegen Datenmodelle, mit denen die Abbildung der Wirklichkeit auf Datenstrukturen und damit die Bedeutung der Daten festgelegt werden.
- Verfahren zur **Datenextraktion** werden in **Kap. 4** beleuchtet. Mit Selektionsverfahren und -sprachen wie regulären Ausdrücken, XPath oder SQL lassen sich unstrukturierte in strukturierte Daten transformieren oder Teildatensätze auswählen, um sie schließlich für die Datenanalyse in Tabellen- oder Matrizenform umzuformen, zusammenzuführen und zu aggregieren.

Im zweiten Teil des Buchs findet sich jeweils eine kurze Einführung in die Programmierung mit R und Python, zwei der wichtigsten Sprachen für die Anwendung von Computational Methods. Diese Sprachen verweisen auf unterschiedliche Traditionen. Während R näher an der Welt der Statistik ist, wird Python insbesondere von Wissenschaftler:innen mit Informatikhintergrund eingesetzt. Beide Sprachen sind sehr gut dazu geeignet, Daten zu erheben, aufzubereiten und auszuwerten. Je nach Anwendungsgebiet können Sie persönliche Vorlieben ausbilden, sodass beispielsweise die Datenerhebung mit Python und die Datenanalyse mit R schneller von der Hand gehen.

- **Abschn. 5.1** führt kurz und knapp in die **Programmierung mit R** ein. Hier lernen Sie zunächst, wie die Entwicklungsumgebung RStudio aufgebaut ist, wie Befehle formuliert und Skripte entwickelt werden. Dabei werden grundlegende Funktionen, unter anderem aus dem Tidyverse, besprochen, um Datensätze einzulesen, zu filtern und zu analysieren. Auch das Erstellen von Grafiken ist ein wesentlicher Bestandteil der Datenanalyse mit R.
- **Abschn. 5.2** bietet eine praxisorientierte Einführung in die **Programmierung mit Python**. Zu Beginn lernen Sie, wie Sie Jupyter-Notebooks einrichten und nutzen. Anschließend wird in Basisbefehle, Datenstrukturen und Funktionen eingeführt. Für die Datenanalyse lernen Sie Funktionen aus der weit verbreiteten Programmbibliothek *pandas* kennen.
- In **Kap. 6** finden Sie Hilfestellungen, sobald Programmierprojekte größer werden, etwa wenn mehrere Personen gleichzeitig an einem Projekt arbeiten oder die Datenmengen sehr umfangreich werden. Dabei erfahren Sie, wann sich die Arbeit mit Versionsverwaltungen lohnt, wie Computer virtualisiert werden können und wie Datenanalysen auf ein Cluster für High-Performance-Computing ausgelagert werden.

Der dritte Teil des Buchs beschäftigt sich schließlich mit konkreten Anwendungsfeldern von Computational Methods, in denen die bis dahin thematisierten Grundtechniken eingesetzt werden. Die Beispiele werden in einer der beiden Programmiersprachen R oder Python angeleitet und gegebenenfalls um Hinweise zur Umsetzung in der jeweils anderen Sprache ergänzt. Jedes der Kapitel fasst die jeweiligen Verfahren kurz zusammen und führt schrittweise durch ein praktisches Beispiel. Zunächst werden Verfahren automatisierter Datenerhebung thematisiert:

- Eine Einführung in **Webscraping**, um Inhalte aus Webseiten auszulesen, bietet **Abschn. 7.1**. Dabei lernen Sie zunächst, wie Sie mit Python einzelne HTML-Dokumente herunterladen, Daten aus dem Quelltext extrahieren und abspeichern. Um mehrere Webseiten abzufragen, kann der Webbrowser mithilfe von Selenium automatisiert werden und Sie finden Hinweise auf Programme und Plattformen, die beim Webscraping unterstützen.
- Wie Sie mit **Application Programming Interfaces (APIs)** arbeiten, lernen Sie in **Abschn. 7.2**. Über APIs lassen sich vorstrukturierte Daten erheben, die von den Plattformbetreibern bereitgestellt werden. Zwei Anwendungsfälle verdeutlichen den Nutzen von APIs: Zum einen werden Social-Media-Daten mithilfe von Facepager über die Twitter-API erhoben und zum anderen wird automatische Bilderkennung über Googles Cloud-Vision-API vorgestellt.

Sobald die Daten vorliegen, kommt es zur Datenanalyse, um inhaltliche Fragestellungen zu beantworten:

- **Vorhersagen** und **Klassifikationen** haben eine lange Tradition in der Statistik, zum Beispiel in der Regressionsanalyse. Sie werden aus Sicht von Computational Methods häufig als Probleme des **Machine Learnings** begriffen und in **Kap. 8** behandelt. Zunächst wird in grundlegende Konzepte des maschinellen Lernens eingeführt. Als überwachtes Lernverfahren wird in Abschn. 8.1 ein künstliches neuronales Netz trainiert, um damit Bilder automatisiert vorgegebenen Kategorien zuzuordnen. In Abschn. 8.2 wird als unüberwachtes Lernverfahren Topic Modelling angewendet, mit dem Texte ohne vorab bekannte Kategorien sortiert werden.
- **Kap. 9** beschäftigt sich mit der automatisierten **Textanalyse**, die Texte als Daten begreift, indem sie diese in ihre Bestandteile zerlegt und in Variablen überführt. Das Kapitel behandelt Grundtechniken zum Auszählen von Wörtern, die diktionärsbasierte Inhaltsanalyse und gibt einen Ausblick auf die Analyse von Syntax und Semantik.

- Die Beziehungen zwischen Akteuren, aber auch Konzepten und Ereignissen, können über **Netzwerkanalysen** betrachtet werden. Aus netzwerkanalytischer Sicht interessiert beispielsweise, wie sich Informationen verbreiten oder wie sich die Ressourcen eines Akteurs durch die Beziehungen zu anderen Akteuren erklären lassen. Das **Kap. 10** beinhaltet eine Einführung in die grundlegenden Konzepte der Netzwerkanalyse sowie ein praktisches Beispiel zur Erhebung, Analyse und Visualisierung eines Netzwerkes.
- **Kap. 11** führt in **Simulationsverfahren** ein, bei welchen hypothetische Welten erschaffen und mit empirisch vorgefundenen Welten verglichen werden. Dadurch lässt sich zum einen nachvollziehen, wie aus dem Verhalten einzelner Akteure auf der Ebene von Gesamtsystemen komplexe Effekte emergieren – etwa inwiefern es zur Fragmentierung von Öffentlichkeit kommt, wenn vor allem personalisierte Inhalte konsumiert werden. Zum anderen kann überprüft werden, inwiefern die in einem Datensatz vorgefundenen Zusammenhänge überzufällig oder auffällig erscheinen, wenn sie mit kontrafaktischen Welten verglichen werden.

Auch wenn die Reihenfolge der drei Teile eine bestimmte Leserichtung nahelegt, sind alle Kapitel so konzipiert, dass sie losgelöst von den anderen gelesen werden können. Wir hoffen, dass sie gleichzeitig als Inspiration und als Nachschlagewerk dienen können. Für einen schnellen Einstieg kopieren Sie die Skriptschnipsel aus dem Buch oder Repositorium und wandeln Sie diese für eigene Zwecke ab. Fehler sind selbstverständlich vorprogrammiert.

Übungsfragen

1. Was versteht man unter Computational Methods?
2. Wählen Sie eine wissenschaftliche Disziplin und finden Sie heraus, wie die Verbindung mit Computational Methods in dieser Disziplin bezeichnet wird! Wie heißt dieser Bereich zum Beispiel in der Musik oder in der Physik?
3. Was sind forschungspraktische Konsequenzen von Automatisierung?
4. Was sind absolute und relative Pfade?
5. Wie öffnen Sie die Kommandozeile?

Weiterführende Literatur

Attewell, P. A. & Monaghan, D. B. (2015). *Data mining for the social sciences. An introduction.* Oakland: University of California Press.

Cioffi-Revilla, C. (2017). *Introduction to computational social science. Principles and applications* (2. Aufl.). London: Springer.

Jannidis, F., Kohle, H. & Rehbein, M. (2017). *Digital Humanities. Eine Einführung*. Stuttgart: J.B. Metzler.

Rogers, R. (2013). *Digital Methods.* Cambridge: The MIT Press.

Sloan, L. & Quan-Haase, A. (Hrsg.). (2017). *The SAGE handbook of social media research methods.* Los Angeles: SAGE reference.

Literatur

Almind, T. C. & Ingwersen, P. (1997). Informetric analyses on the world wide web: methodological approaches to 'webometrics'. *Journal of Documentation*, *53*(4), 404–426. https://doi.org/10.1108/EUM0000000007205

Amaral, I. (2017). Computational Social Sciences. In L. A. Schintler & C. L. McNeely (Hrsg.), *Encyclopedia of Big Data* (S. 1–3). Cham: Springer. https://doi.org/10.1007/978-3-319-32010-6_41

Attewell, P. A. & Monaghan, D. B. (2015). *Data mining for the social sciences. An introduction*. Oakland: University of California Press.

Beck, K. (2006). *Computervermittelte Kommunikation im Internet*. München: Oldenbourg. https://doi.org/10.1524/9783486839203

Björneborn, L. (2004). *Small-world link structures across an academic web space. A library and information science approach*. Copenhagen: Royal School of Library and Information Science.

Björneborn, L. & Ingwersen, P. (2004). Toward a basic framework for webometrics. *Journal of the American Society for Information Science and Technology*, *55*(14), 1216–1227. https://doi.org/10.1002/asi.20077

Briatte. (2021). *Awesome Network Analysis. An awesome list of resources to construct, analyze and visualize network data*. Zugriff am 19.04.2022. https://github.com/briatte/awesome-network-analysis

Burton, O. V. (2005). American Digital History. *Social Science Computer Review*, *23*(2), 206–220. https://doi.org/10.1177/0894439304273317

Busa, R. A. (2004). Perspectives on the Digital Humanities. In S. Schreibman, R. G. Siemens & J. Unsworth (Hrsg.), *A companion to digital humanities* (S. xvi–xxi). Oxford: Blackwell.

Cao, L. (2017). Data Science. *ACM Computing Surveys*, *50*(3), 1–42. https://doi.org/10.1145/3076253

Cioffi-Revilla, C. (2010). Computational social science. *Wiley Interdisciplinary Reviews: Computational Statistics*, *2*(3), 259–271. https://doi.org/10.1002/wics.95

Cioffi-Revilla, C. (2017). *Introduction to computational social science. Principles and applications* (2. Aufl.). London: Springer.

CLARIAH-DE. (2022). *Willkommen bei CLARIAH-DE*. Zugriff am 16.05.2022. https://www.clariah.de/

Cohen, D. J. & Rosenzweig, R. (2006). *Digital history. A guide to gathering, preserving, and presenting the past on the Web*. Philadelphia: University of Pennsylvania Press.
Coleman, J. S. (1964). *An Introduction to Mathematical Sociology*. New York: Free Press.
Deutsche Gesellschaft für Publizistik- und Kommunikationswissenschaft (2022). https://www.dgpuk.de/de/forschungssoftware.html
DHI. (2020). *The DHI and Digital Humanities*. Zugriff am 08.05.2020. https://www.ceu.edu/dhi/what-is-digital-humanities
Fayyad, U., Piatetsky-Shapiro, G. & Smyth, P. (1996). From Data Mining to Knowledge Discovery in Databases. *AI Magazine*, *17*(3), 37–54. https://doi.org/10.1609/aimag.v17i3.1230
Git. (2022a). Git. Fast-version-control (Version 2.36.0) [Computer software]. https://git-scm.com/downloads
GitHub. (2022a). Atom (Version 1.60.0) [Computer software]. https://atom.io/
Ho, D. (2022). Notepad++ (Version 8.4.1) [Computer software]. https://notepad-plus-plus.org/
ICA CM. (2017). *About the ICA Computational Methods Interest Group*. Zugriff am 29.11.2019. http://ica-cm.org
Jannidis, F., Kohle, H. & Rehbein, M. (2017). *Digital Humanities. Eine Einführung*. Stuttgart: J.B. Metzler.
Jünger, J. (2018). Mapping the field of automated data collection on the web. Data types, collection approaches and their research logic. In C. M. Stützer, M. Welker & M. Egger (Hrsg.), *Computational social science in the age of big data. Concepts, methodologies, tools, and applications* (S. 104–130). Köln: Halem.
Jünger, J. & Schade, H. (2018). Liegt die Zukunft der Kommunikationswissenschaft in der Vergangenheit? Ein Plädoyer für Kontinuität statt Veränderung bei der Analyse von Digitalisierung. *Publizistik*, *63*(4), 497–512. https://doi.org/10.1007/s11616-018-0457-6
MacroMates. (2021). TextMate (Version 2.0) [Computer software]. https://macromates.com/
Marres, N. (2017). *Digital sociology. The reinvention of social research*. Cambridge: Polity.
Microsoft. (2022b). Visual Studio Code (Version 1.67.1) [Computer software]. https://code.visualstudio.com/
Monroe, B. L. & Schrodt, P. A. (2008). Introduction to the special issue: The statistical analysis of political text. *Political Analysis*, *16*(4), 351–355. https://doi.org/10.1093/pan/mpn017
Munroe, R. (2013). *Automation. xkdc: A webcomic of romance, sarcasm, math, and language.* Zugriff am 16.05.2022. https://xkcd.com/1319/
Rogers, R. (2010). Internet Research. The Question of Method. A Keynote Address from the YouTube and the 2008 Election Cycle in the United States Conference. *Journal of Information Technology & Politics*, *7*(2–3), 241–260. https://doi.org/10.1080/19331681003753438
Russell, S. J. & Norvig, P. (2012). *Künstliche Intelligenz. Ein moderner Ansatz* (3., aktual. Aufl.). München: Pearson.
Scharkow, M. (2011). Zur Verknüpfung manueller und automatischer Inhaltsanalyse durch maschinelles Lernen. *Medien & Kommunikationswissenschaft*, *59*(4), 545–562. https://doi.org/10.5771/1615-634x-2011-4-545
Schubert, L. (2020). Computational Linguistics. In E. N. Zalta (Hrsg.), *The Stanford Encyclopedia of Philosophy.* Zugriff am 01.04.2020. https://plato.stanford.edu/archives/spr2020/entries/computational-linguistics/

Shah, D. V., Cappella, J. N. & Neuman, W. R. (2015). Big Data, Digital Media, and Computational Social Science. *The ANNALS of the American Academy of Political and Social Science*, *659*(1), 6–13. https://doi.org/10.1177/0002716215572084

Social Media Data Stewardship. (2021). *Social Media Research Toolkit*. Zugriff am 16.05.2022. https://socialmediadata.org/social-media-research-toolkit/

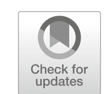
Stack Overflow. (2022). *A public platform building the definitive collection of coding questions & answers*. Zugriff am 24.04.2022. https://stackoverflow.com

Terras, M., Nyhan, J. & Vanhoutte, E. (Hrsg.). (2013). *Defining digital humanities. A reader*. Farnham: Ashgate.

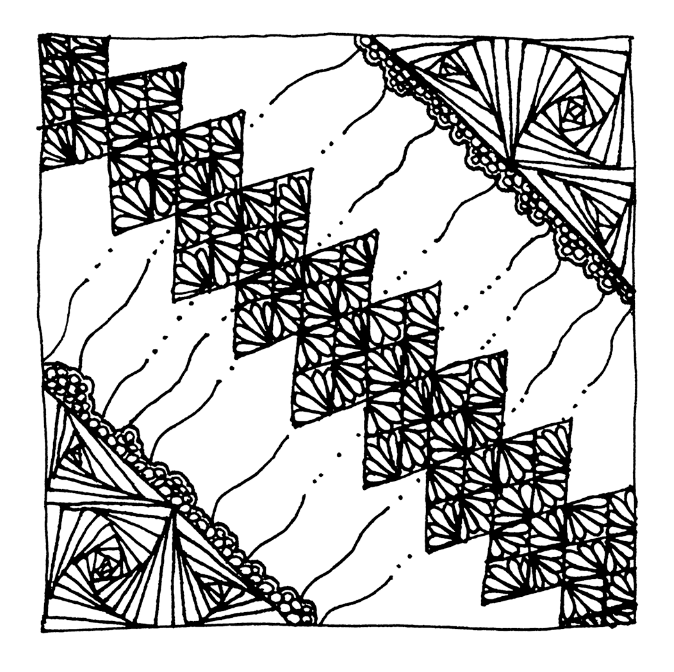
Thelwall, M. [Michael]. (2009). *Introduction to Webometrics. Quantitative Web Research for the Social Sciences*. San Rafael: Morgan & Claypool. https://doi.org/10.2200/S00176ED1V01Y200903ICR004

Van Atteveldt, W. & Peng, T.-Q. (2018). When Communication Meets Computation: Opportunities, Challenges, and Pitfalls in Computational Communication Science. *Communication Methods and Measures*, *12*(2–3), 81–92. https://doi.org/10.1080/19312458.2018.1458084

Welker, M. (2019). Computer- und onlinegestützte Methoden für die Untersuchung digitaler Kommunikation. In W. Schweiger & K. Beck (Hrsg.), *Handbuch Online-Kommunikation* (S. 531–572). Wiesbaden: Springer Fachmedien. https://doi.org/10.1007/978-3-658-18016-4_21



2 Datenquellen



J. Jünger, C. Gärtner, *Computational Methods für die Sozial- und Geisteswissenschaften*, https://doi.org/10.1007/978-3-658-37747-2_2

Zusammenfassung

Dieses Kapitel beinhaltet eine Einführung in Datenquellen, aus denen Daten mit Computational Methods gewonnen und analysiert werden können. Sie lernen, worin sich verschiedenen Verfahren automatisierter Datenerhebung unterscheiden und wie Ressourcen im Web identifiziert werden. Außerdem erfahren Sie, wo wissenschaftlich verwertbare Daten zu finden sind.

Im Online-Repositorium unter https://github.com/strohne/cm finden Sie begleitend zum Kapitel weitere Materialien, auf die wir im Text mit ☛ verweisen.

Schlüsselwörter

Uniform Resource Locator (URL) · Application Programming Interface (API) · Repositorien · Datenspenden · Open Data

Im Bereich Computational Methods besteht eine zentrale Frage darin, wie automatisiert sozial- und geisteswissenschaftlich relevante Daten gewonnen werden können. Automatisierte Datenerhebungen sind vor allem dort sinnvoll, wo sehr viele Daten anfallen, die man ungern manuell aufbereiten will. Das trifft beispielsweise für die Kommunikation auf Online-Plattformen wie Facebook zu. Die Daten sind dabei in der Regel schon vorhanden und werden nicht erst von Wissenschaftler:innen erstellt. Sie fallen im Zusammenhang mit menschlichem Verhalten ohnehin an, deshalb spricht man auch von prozessgenerierten Daten (Johnson und Turner 2003). Dennoch kann man nicht davon ausgehen, dass hier ein unverfälschter Zugang zu einer ohnehin vorhandenen sozialen Wirklichkeit besteht, vielmehr wird Wirklichkeit erst auf den Plattformen erzeugt – zum Beispiel spielen die Funktionen der Plattformen eine wichtige Rolle dafür, welche Daten entstehen und zugänglich sind (siehe zum Beispiel Jünger 2021).

Über verschiedene Datenzugänge werden unterschiedliche Repräsentationen von Wirklichkeit sichtbar. Dieser Punkt lässt sich gut am Beispiel von Online-Kommunikation verdeutlichen. Aus technischer Sicht ist Online-Kommunikation dadurch gekennzeichnet, dass zwei Maschinen miteinander interagieren. Auf der Seite der Nutzer:innen wird diese Maschine als Client bezeichnet, der Client schickt eine Anfrage an einen Server. Der Server bearbeitet die Anfrage und schickt eine Antwort zurück. Beim Surfen im Web findet dieses Wechselspiel ausgehend von einem Browser wie Firefox, Chrome oder Safari statt. Automatisierte Datenerhebung ist nun dadurch gekennzeichnet, dass Skripte oder Programme eingesetzt

werden, um Daten beim Server anzufragen. Statt also die Adresse https://www.google.de in den Browser einzugeben, wird diese Adresse in einem Erhebungstool erzeugt und das Ergebnis wird weiterverarbeitet (Abb. 2.1).

Normalerweise antworten Webserver mit HTML-Dateien, die dann im Browser grafisch dargestellt werden. Diese HTML-Dateien enthalten die Daten und Verweise auf weitere Dateien, die zur Darstellung benötigt werden, etwa zu Bilddateien. Formatvorlagen in der Form von CSS-Dateien steuern darüber hinaus die Gestaltung, etwa die Schriftfarbe, und JavaScript-Dateien fügen interaktive Elemente hinzu, zum Beispiel zum Auf- und Zuklappen von Menüs. Wenn bei der automatisierten Datenerhebung mit diesen HTML-Dateien gearbeitet wird, spricht man von Webscraping (siehe Abschn. 7.1). Die Dateien werden hierbei nicht angezeigt, sondern es werden einzelne Daten wie Texte oder Tabellen aus dem HTML-Quelltext von Webseiten extrahiert.

Viele Webseiten, unter anderem Social-Media-Plattformen, stellen zusätzlich sogenannte Application Programming Interfaces (APIs) zur Verfügung. Eine solche API unterscheidet sich von Webseiten dadurch, dass sie für den automatisierten Zugriff unabhängig von einem Browser entwickelt wird. Während Anbieter die Struktur einer Webseite bei Bedarf unangekündigt ändern – vor allem, wenn neue Funktionen eingeführt werden –, garantieren die Betreiber von APIs in der Regel,

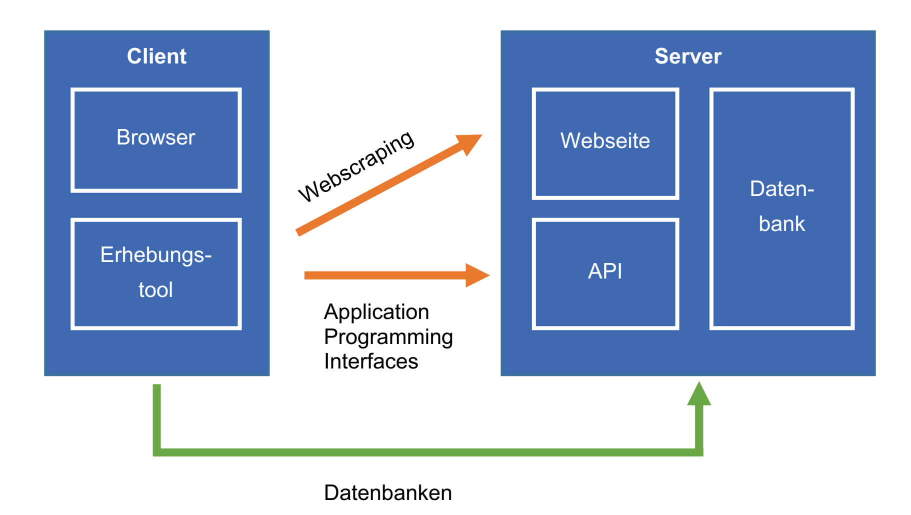


Abb. 2.1 Verfahren automatisierter Datenerhebung im Web. (Quelle: eigene Darstellung)

dass diese über lange Zeiträume stabil bleiben. Nur deshalb lohnt sich für Drittanbieter die Investition in eigene Apps, die auf Fremddaten aufbauen. Würde sich etwa die Struktur der Google-Maps-API immer wieder ändern, gäbe es keine Garantie, dass eine darauf aufbauende Geocaching-App danach noch funktioniert. Ein weiterer Unterschied besteht darin, dass die zurückgegebenen Daten nicht im HTML-Format, sondern in stärker vorstrukturierten und damit leichter verarbeitbaren Formaten wie JSON verschickt werden (siehe Kap. 3).

Sowohl der Zugang über Webscraping als auch die Nutzung von APIs sind davon abhängig, welche Daten ein Betreiber zugänglich macht. Im Endeffekt vermitteln beide Wege den kontrollierten Zugriff auf die Datenbanken des Anbieters, es handelt sich dabei jedoch um verschiedene Repräsentationen der Daten. In seltenen Fällen werden die Datenbanken selbst zur Verfügung gestellt. Ein Beispiel wäre die Wikipedia-Datenbank. Auch hier erfolgt der Zugriff aber nicht unvermittelt, zum einen muss die Datenbank erst heruntergeladen werden und zum anderen ist für die Arbeit mit Datenbanken ein passendes Datenbankmanagementsystem (DBMS) nötig (siehe Kap. 3). Insofern gibt es keine unvermittelten Daten und es muss stets reflektiert werden, wofür die erhobenen Daten stehen. In den folgenden Kapiteln werden Grundlagen zu den drei verschiedenen Datenzugängen und entsprechenden Datenformaten vermittelt sowie einige typische Datenquellen benannt.

2.1 Webseiten

Ein wesentliches Element des Web sind Uniform Resource Locators (URLs), mit denen die verschiedenen Ressourcen im Web adressiert werden. Eine solche URL besteht in der Regel aus fünf Komponenten (Abb. 2.2, ☛ *Repositorium*):

- Das **Protokoll** gibt an, in welcher Sprache die beiden Computer miteinander interagieren. Im Web wird dafür typischerweise HTTP (Hypertext Transfer Protokoll) oder für verschlüsselte Kommunikation HTTPS verwendet.
- Die **Domain** identifiziert den Server und besteht selbst wieder aus mehreren durch Punkt getrennten Teilen. Im Beispiel ist die Top-Level-Domain „com“, die Domain selbst ist „youtube“ und das Präfix „www“ wird als Subdomain bezeichnet.
- Der **Pfad** gibt die Operation oder die Ressource auf dem Server an. Vereinfachend kann man davon ausgehen, dass hinter einer Domain ein Computer steht und der Pfad die Datei oder das Programm angibt, auf welche zugegriffen werden soll.

- Nach einem Fragezeichen folgen die **Parameter der Anfrage** (engl. *query string*), um dem Server genauere Angaben zum Auffinden der Ressource mitzugeben. Ein Parameter besteht immer aus einem Namen, auf den nach einem Gleichheitszeichen ein Wert folgt. Mehrere Parameter werden durch das &-Zeichen getrennt.
- Das **Hashfragment** am Ende der URL wird niemals an den Server gesendet, sondern lediglich im Browser ausgewertet. Damit werden einzelne Teile der Webseite angesprochen, zum Beispiel der Kommentarbereich, sodass die Anzeige direkt zu diesem Abschnitt springen kann.

Darüber hinaus können in Internetadressen noch weitere Angaben vorkommen, die für die Bearbeitung der Anfrage relevant sind, beispielsweise ein sogenannter Port oder Zugangsdaten.

Zu beachten ist, dass einige Zeichen in URLs speziell kodiert werden müssen – das betrifft die für den Aufbau der URL reservierten Zeichen wie das Fragezeichen, aber auch Umlaute. Diese Zeichen werden durch Prozentkodierung angegeben, sodass aus dem Umlaut ä beispielsweise %C3%A4 wird. Für das Leerzeichen gibt es zwei Varianten, zum einen kann universell die Prozentkodierung %20 verwendet werden, zum anderen ist innerhalb der Parameter das Pluszeichen + anzutreffen.

Welche Rolle spielen URLs nun für Daten auf Webseiten? Im einfachsten Fall ist eine Ressource durch eine URL abrufbar und die entsprechende Webseite enthält Daten wie zum Beispiel eine eingebettete Tabelle mit Mitgliederzahlen politischer Parteien. Auch die gesamte Webseite kann Gegenstand der Analyse sein, wenn Blogs oder Nachrichtenseiten untersucht werden sollen. Zudem sind die URLs selbst für wissenschaftliche Analysen von Interesse, denn dadurch können die Verbindungen zu anderen Seiten verfolgt werden, um so Netzwerke zwischen Webseiten und Akteuren nachzuvollziehen.

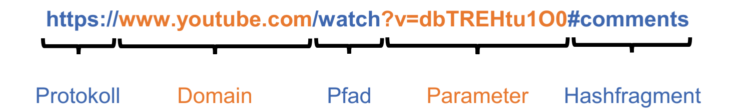


Abb. 2.2 Die Bestandteile einer URL. (Quelle: eigene Darstellung)

Im Web findet sich eine Vielzahl an Webseiten, auf denen Daten beispielsweise in Tabellen- oder Listenform bereitgestellt werden:

- Primärdaten finden sich insbesondere in Registern politisch-administrativer Angebote. Eine Anlaufstelle für Listen von Unternehmen oder Vereinen ist in Deutschland das gemeinsame Registerportal der Länder (Ministerium der Justiz Nordrhein-Westfalen 2020). Auch das statistische Bundesamt stellt Daten zur Verfügung (Destatis 2022).
- Berufsverbände und andere Interessenvertretungen listen häufig Daten über ihre Mitglieder auf. So lassen sich Medienangebote unter anderem über die Informationsgesellschaft zur Feststellung der Verbreitung von Werbeträgern (IVW 2022) oder über den Bundesverband Digitalpublisher und Zeitungsverleger (BDZV 2022) identifizieren.
- Themenportale und Plattformen bieten häufig Überblicksseiten über ihre Datenbestände an. Das reicht von Medienangeboten wie Fernsehserien oder Podcasts über Fußballergebnisse bis zu Cocktailrezepten und App-Stores.
- Aufbereitete Daten zu allen möglichen Themen finden sich auch in den Artikeln von Online-Enzyklopädien und Nachschlagewerken wie Wikipedia (Wikimedia Deutschland 2022) oder Fandom (Fandom 2022).

Beim Zugang zu diesen Daten stößt man auf ganz unterschiedliche Rahmenbedingungen. Im einfachsten Fall sind alle Daten auf einer einzelnen über eine URL identifizierbare Seite enthalten, zum Beispiel in einer Wikipedia-Tabelle (siehe Abschn. 7.1). Häufig werden lange Listen aber auch über mehrere Seiten verteilt, wobei jede Seite eine eigene URL erhält. Beobachten Sie beim Surfen im Web die URLs: Typischerweise wird die Seite über einen Parameter wie `page=5` angegeben und Protokoll, Domain sowie Pfad bleiben gleich.

Die Paginierung, das heißt die Aufteilung auf mehrere Seiten, wird auf stark interaktiven Seiten jedoch nicht immer in der Adressleiste sichtbar, sondern die einzelnen Seiten werden beim Scrollen nach und nach über JavaScript und sogenannte XMLHttpRequests nachgeladen, was die automatisierte Erhebung erschwert. Eine weitere Hürde sind Datenbanken, in denen über Suchformulare recherchiert wird. Hier reicht die Angabe einer URL nicht aus, sondern die Suchbegriffe müssen auf anderem Weg an den Server übermittelt werden. Das HTTP-Protokoll sieht verschiedene Methoden der Interaktion mit einem Webserver vor: GET-Anfragen rufen eine URL auf, POST-Anfragen senden weitere Nutzdaten an diese Adresse und DELETE-Anfragen sind zum Löschen von Daten vorgesehen. Suchfunktionen bauen häufig auf POST-Anfragen auf. Da die Daten nicht über Links identifizierbar sind, können solche Inhalte auch nicht einfach über

Suchmaschinen wie Google gefunden werden, wofür sich der Begriff Deep Web eingebürgert hat.

Welche Anfragen genau beim Surfen an einen Server geschickt werden, lässt sich gut mit der Entwicklerkonsole des Browsers nachvollziehen, die in den meisten Browsern mit der Taste F12 aktiviert wird (siehe das Beispiel zum Extrahieren von URLs in Abschn. 4.1). Der praktische Umgang mit den verschiedenen Formen von Webscraping wird in Abschn. 7.1 vermittelt. An dieser Stelle sind zunächst drei Hinweise auf ethisch-rechtliche Voraussetzungen wichtig. Erstens enthalten die Nutzungsbedingungen (engl. *terms of services*) von Webseiten und insbesondere von Social-Media-Plattformen Regelungen, mit denen sich die Betreiber eine automatisierte Datenerhebung häufig verbitten. Allerdings gehören insbesondere Suchmaschinen wie Google, die solche Erhebungen systematisch durchführen, selbstverständlich zum Web dazu, diese erfassen über automatisiertes Crawling die Inhalte von anderen Webseiten. Einen Einblick, welche Regelungen eine Webseite dafür vorsieht, können Sie sich über die robots.txt verschaffen. Diese Datei ist in der Regel auf jedem Server verfügbar und kann abgerufen werden, indem der Name an den Domainnamen angehängt wird, probieren Sie es zum Beispiel bei Facebook aus: https://www.facebook.com/robots.txt. Zweitens finden sich in der Datenschutzgrundverordnung und im Urheberrecht spezielle Regelungen für wissenschaftliche Zwecke, die zum Beispiel Textmining unter bestimmten Umständen explizit erlauben. Drittens sind ethische Abwägungen notwendig, insbesondere, wenn personenbezogene Daten betroffen sind. Das bedeutet: Die Möglichkeiten und Grenzen automatisierter Erhebungen müssen für jedes Projekt im Einzelfall reflektiert werden. Orientierung geben Ethikkodizes und Handreichungen (zum Beispiel RatSWD 2019) und der rege Diskurs in der Forschungsliteratur (unter anderem Bruns 2019; Fiesler et al. 2020; Kotsios et al. 2019; Thelwall und Stuart 2006).

2.2 Application Programming Interfaces

Das Extrahieren von Daten auf Webseiten baut zwar auf Standards im Web auf, erfordert jedoch mitunter eine detaillierte Auseinandersetzung mit der Struktur der Webseiten. Zudem ist Webscraping besonders von einigen Social-Media-Plattformen wie Facebook, Twitter oder YouTube laut deren Nutzungsbedingungen nicht erwünscht. Webseitenbetreiber bauen mitunter Hürden ein, um das Webscraping zu erschweren. Ein bereits vorstrukturierter Datenzugang ist aber bei vielen Plattformen mittels Application Programming Interfaces (APIs) möglich, über die Webseitenbetreiber kontrollieren, wer wie viele und welche Daten erheben kann.

Ganz allgemein legen Programmierschnittstellen fest, wie zwei Programme miteinander interagieren können (Jacobson et al. 2012, S. 5). Diese Schnittstellen sind meistens nicht vorrangig für wissenschaftliche Datenanalysen eingerichtet worden, sondern für die Entwicklung von Drittanwendungen. Im Web wird auf diese Weise beispielsweise die Funktion umgesetzt, dass man sich auf anderen Seiten „Mit Google einloggen" kann. Die API-Anbieter legen dazu Endpunkte und Parameter fest, auf die andere Programme, sogenannte API-Konsumenten, zugreifen. Ein Endpunkt ist einfach eine URL wie `https://api.twitter.com/2/users/`. Diese URL wird um weitere Pfad- und Queryparameter ergänzt, mit denen etwa die öffentlichen Profilinformationen einzelner Nutzer:innen abgefragt werden. Pfadparameter wie `show.json` werden direkt an den Pfad angehängt, wohingegen Queryparameter wie `?screen_name=wissen_lockt` als Liste von Name-Wert-Paaren nach einem Fragezeichen angegeben werden. Im Gegensatz zu einer normalen Webseite geben APIs in der Regel nicht HTML, sondern deutlich leichter verarbeitbare JSON-Formate zurück (Abb. 2.3 und Kap. 3).

Eine API ist mehr als eine Software, sie ist ein Vertrag zwischen dem API-Anbieter und dem API-Konsumenten. Der Anbieter sichert damit zu, dass der Zugriff über einen längeren Zeitraum bestehen bleibt und der Konsument sorgt letztendlich dafür, dass sich die Dienste des Anbieters in der Welt verbreiten. So wie sich Hersteller von USB-Sticks darauf verlassen, dass die USB-Buchse immer die gleichen Abmessungen haben, verlassen sich Anwendungsentwickler darauf, dass sich die Endpunkte und die Datenformate nicht ändern. Wichtig ist deshalb die genaue Dokumentation (engl. *reference)* der API, in der Endpunkte, Parameter und Rückgabeformate beschrieben werden. Die Betreiber stellen die Dokumentation mehr oder weniger übersichtlich zusammen, darüber hinaus kommen auch maschinenlesbare Standards wie OpenAPI[1] zum Einsatz.

Viele für wissenschaftliche Analysen verwendete APIs bauen auf REST-Prinzipien (Fielding 2000) auf, das heißt, einzelne Ressourcen sind wie Webseiten über URLs ansprechbar. Einige APIs sind so weit standardisiert, dass die Endpunkte immer auf die gleiche Art und Weise aufgebaut sind oder dass auch die Dokumentation der API über die API selbst abgerufen werden kann. Beispielsweise folgen die zum Abgleich heterogener Datenbestände eingesetzten Reconci-

[1] Siehe OpenAPI (2022; https://github.com/OAI/OpenAPI-Specification).

URL der Webseite:

https://twitter.com/wissen_lockt

```
▼<div class="css-1dbjc4n r-18u37iz"> flex
 ▶<div class="css-1dbjc4n">…</div> flex
 ▼<div class="css-1dbjc4n r-1joea0r"> flex
  ▼<a class="css-4rbku5 css-18t9o4 css-901oao r-jwli3a r-1loqt21 r-1qd0xha r-a023e6 r-16dba41 r-ad9z0x r-bcqeeo r-qvutc0"
   title="3,119" href="/wissen_lockt/followers" dir="auto" role="link" data-focusable="true"> event
   ▼<span class="css-901oao css-16my406 r-1qd0xha r-vw2c0b r-ad9z0x r-bcqeeo r-qvutc0">
      <span class="css-901oao css-16my406 r-1qd0xha r-ad9z0x r-bcqeeo r-qvutc0">3,119</span>
    </span>
   ▼<span class="css-901oao css-16my406 r-111h2gw r-1qd0xha r-ad9z0x r-bcqeeo r-qvutc0">
      <span class="css-901oao css-16my406 r-1qd0xha r-ad9
    </span>
   </a>
  </div>
</div>
```

Anzahl der Follower eines Twitter-Accounts im HTML-Quelltext

URL des API-Endpunkts:

https://api.twitter.com/1.1/users/show.json?screen_name=wissen_lockt

```
"name": "Uni Greifswald",
"screen_name": "wissen_lockt",
"location": "Greifswald",
"profile_location": null,
"description": "Hier twittert di                    UniGreifswald!\n
"protected": "False",
"followers_count": "3119",
"friends_count": "349",
"listed_count": "87",
"created_at": "Fri Apr 30 13:06:27 +0000 2010",
```

Anzahl der Follower eines Accounts in der JSON-Antwort der Twitter-API

Abb. 2.3 Inhalt einer Webseite (HTML) und Antwort einer API (JSON) im Vergleich. (Quelle: eigene Darstellung)

liation Service APIs[2] einer übergeordneten Spezifikation und können so in Tools wie OpenRefine[3] verwendet werden. OpenRefine ermöglicht es, über die Einbindung mehrerer APIs etwa eine Liste von Personen oder Unternehmen gleichzeitig mit einem Register politisch sanktionierter Akteure und mit Einträgen bei der Deutschen Nationalbibliothek abzugleichen.[4] Auch die im Semantic Web einge-

---

[2] Siehe Entity Reconciliation Community Group (2022; https://reconciliation-api.github.io/specs/0.1/).

[3] Siehe Huynh (2022; https://openrefine.org/).

[4] Für eine Liste von APIs, die in OpenRefine eingebunden werden können, siehe OpenRefine (2021; https://github.com/OpenRefine/OpenRefine/wiki/Reconcilable-Data-Sources).

setzten APIs folgen Standards wie Hydra[5] oder stellen nach einem festgelegten Schema sogenannte SPARQL-Endpunkte bereit (siehe Kap. 3). Trotz dieser Standardisierungen bleibt eine Auseinandersetzung mit den speziellen Endpunkten einer API nicht aus.

Wie auch auf Webseiten setzen viele Anbieter eine Registrierung oder sogar eine Vorabprüfung des geplanten Projekts voraus. Insbesondere bei Social-Media-Plattformen wie Facebook oder YouTube ist der Zugang stark kontrolliert. Dagegen setzen sich Organisationen wie die Open Knowledge Foundation dafür ein, vor allem Daten öffentlich-rechtlicher Einrichtungen offen zugänglich zu machen, zum Beispiel über das Portal OffeneRegister.de. Diese Bestrebungen werden unter dem Schlagwort Open Data auch politisch aufgegriffen, im Jahr 2017 wurde vom Bundestag dazu das sogenannte Open-Data-Gesetz beschlossen (EGovG §12).

Ein Verzeichnis von APIs und weitere Erläuterungen dazu, was eine API ist, finden Sie auf ProgrammableWeb.[6] Darüber hinaus lohnt es sich stets zu prüfen, ob eine Webseite oder Plattform eine API anbietet, auch wenn dies nicht auf den ersten Blick erkennbar ist. Die Übergänge zwischen Webseiten und APIs sind fließend, weil Webanwendungen in vielen Fällen auf Ebene des dahinter liegenden Content Management Systems auf APIs aufbauen. So reicht es mitunter aus, einen Parameter in der URL einer Webseite zu ändern, um das Format von HTML auf JSON umzustellen (siehe die Übungsaufgabe am Ende des Kapitels).

APIs können für vielfältige geistes- und sozialwissenschaftliche Zwecke eingesetzt werden. Erstens stellen Social-Media-Dienste APIs bereit, mit denen die auf der Plattform erzeugten Inhalte (Posts, Kommentare, …) erhoben werden können. Daneben finden sich zweitens Dienste, die andere Daten sammeln und aggregieren, in Zitationsdatenbanken werden etwa die Literaturverweise von wissenschaftlichen Aufsätzen gesammelt. Drittens stellen Cloud-Computing-Anbieter wie Amazon, IBM, Google oder Microsoft über APIs Analysemöglichkeiten zum Beispiel zur automatisierten Bilderkennung bereit. Einen Überblick über einige APIs finden Sie in Tab. 2.1. Wie Sie selbst mit APIs arbeiten können, wird in Abschn. 7.2 erläutert.

[5] Siehe Hydra W3C Community Group (2018; http://www.hydra-cg.com/drafts/use-cases/2.api-documentation.md).

[6] Siehe Berlind (2022; https://www.programmableweb.com/).

Tab. 2.1 Beispiele für Anbieter von Application Programming Interfaces (APIs)

Anbieter	Datenbestände bzw. mögliche Einsatzbereiche
Plattformdaten	
Facebook API	Posts und Kommentare auf Facebook (Meta 2022; https://developers.facebook.com)
MediaWiki Action API	Daten der Wikimedia-Projekte, z. B. Wikipedia (MediaWiki 2022; https://www.mediawiki.org/wiki/API)
Twitter API	Tweets und Informationen zu Nutzer:innen (Twitter 2022d; https://developer.twitter.com/)
YouTube Data API	Videoinformationen und Kommentare auf YouTube (Google Developers 2022; https://developers.google.com/youtube/)
Aggregation von Daten	
Abgeordnetenwatch	Daten zu Wahlen und Abgeordneten (abgeordnetenwatch.de 2022; https://www.abgeordnetenwatch.de/api)
DWDS	Worthäufigkeiten und Wörterbucheinträge des Digitalen Wörterbuchs der deutschen Sprache (Berlin-Brandenburgische Akademie der Wissenschaften 2022; https://www.dwds.de/d/api)
GovData	Amtliche Daten, beispielsweise zu den Kategorien Bevölkerung, Gesundheit, Verkehr, Justiz (GovData 2022; https://www.govdata.de/)
Media Cloud	Online-Berichterstattung (Media Cloud 2022; https://mediacloud.org/)
OffeneRegister.de	Unternehmensdaten aus dem Handelsregister (Datasette 2022; https://db.offeneregister.de/openregister)
Open Citations	Zitationsdatenbank mit bibliografischen Informationen (Peroni und Daquino 2020; https://opencitations.net/index/api/v1)
Open Corporates	Weltweit gesammelte Unternehmensdaten (OpenCorporates 2022; https://api.opencorporates.com/)
Datenaufbereitung oder -analyse	
Amazon Web Services: Rekognition	Bild- und Videoanalyse (Amazon Web Services 2022a; https://aws.amazon.com/de/rekognition/)
Google Cloud Vision API	Erkennen von Objekten auf Bildern (Google 2022d; https://cloud.google.com/vision)
IBM Watson Machine Learning	Bild- und Textklassifikation (IBM 2022; https://www.ibm.com/de-de/cloud/machine-learning)
Microsoft Cognitive Services	Umwandlung von Audio- in Textdateien und Bilderkennung (Microsoft 2022a; https://docs.microsoft.com/en-us/azure/cognitive-services)
Perspective API	Analyse von Texten auf Hate-Speech (Perspective 2021; https://www.perspectiveapi.com/)

Quelle: Eigene Darstellung

2.3 Datenbanken und Datensätze

Sowohl beim Besuchen von Webseiten als auch beim Einsatz von webbasierten APIs wird der Zugang zu Datenbanken über Schnittstellen vermittelt, die auf dem HTTP-Protokoll aufbauen. Jede Anfrage gibt dabei einen kleinen Ausschnitt der Datenbank zurück. Für wissenschaftliche Studien finden sich im Web auch vollständige Datenbanken. Die Vollständigkeit hat aber ihren Preis: Die Dateien können sehr groß werden und sind nach der internen Logik des Anbieters strukturiert. Ein eindrucksvolles Beispiel ist die Global Database of Events Language and Tone.[7] In diesem Projekt werden im Viertelstundentakt weltweit Nachrichtenseiten mit automatischer Textanalyse ausgewertet, aggregiert und die Ergebnisse werden zum Download zur Verfügung gestellt. Für ein Jahr umfasst die Datenbank über zwei Terrabyte. Die Arbeit mit solch umfangreichen Datensätzen setzt spezifische Kenntnisse zum Umgang mit Datenbanken und auf mehrere Computer verteilte Systeme (Cloud Computing, siehe Abschn. 6.4) voraus. Doch nicht immer muss es sich um solche Datenmengen handeln, auch viele kleinere Datenbanken sind für sozial- und geisteswissenschaftliche Analysen hilfreich. Linguistische Korpora mit Chats oder WhatsApp-Nachrichten oder auch Listen mit den Einwohnerzahlen aller Länder der Welt sind vergleichsweise klein.

Zum Auffinden von Datensätzen eignen sich **Suchmaschinen** wie die Google Dataset Search oder Portale wie Kaggle (Tab. 2.2). Teilweise stellen Organisationen und Online-Plattformen ihre **Datenbanken** ganz oder in Teilen zur Verfügung, etwa die Wikipedia oder auch die International Movie Database. Auch Facebook macht ausgewählte Teile seiner Datenbanken für Wissenschaftler:innen verfügbar, zum Beispiel die meistgeteilten Links (URL dataset). Der Zugang ist in diesem Fall stark restringiert und erfolgt über die Organisationen Social Science One[8] oder Crowdtangle.[9] Eine besonders herausfordernde Datensorte stellen organisationsinterne **Verhaltensdaten** dar, sie umfassen beispielsweise Logdateien der Webseitennutzung, das Kaufverhalten in Online-Shops, die Bibliotheksnutzung oder die Daten von Fitness-Trackern. Der Zugang ist auf Mitarbeitende in den entsprechenden Organisationen bzw. Kooperationspartner beschränkt und nur mit starken datenschutzrechtlichen Schutzmaßnahmen möglich.

Zudem werden die Forschungsdaten wissenschaftlicher Studien zunehmend in öffentlichen **Repositorien** abgelegt, um eine Nachnutzung und Nachprüfung zu er-

[7] Siehe Leetaru (2021; https://gdeltproject.org/).

[8] Siehe Harvard University (2022b; https://socialscience.one/).

[9] Siehe Meta (2021; https://www.crowdtangle.com/).

Tab. 2.2 Beispiele für online verfügbare Datenbanken

Suchmaschinen, Repositorien und Verzeichnisse	
AMiner	Datensätze für soziale Netzwerkanalysen (CKCEST 2022; https://cn.aminer.org/data-sna)
CLARIN Virtual Language Observatory	Suchmaschine für Sprachdaten (CLARIN 2022; https://vlo.clarin.eu/)
GESIS	Repositorium für sozialwissenschaftliche Forschungsdaten (GESIS 2022; https://search.gesis.org)
Google Dataset Search	Suche nach öffentlichen Datensätzen (Google 2022f; https://datasetsearch.research.google.com)
Google Public Data	Verzeichnis mit Datensätzen (Google 2014; https://www.google.com/publicdata/directory)
Harvard Dataverse	Repositorium für Forschungsdaten (Harvard University 2022a; https://dataverse.harvard.edu/)
ICPSR datasets	Repositorium für Forschungsdaten (University of Michigan 2022; https://www.icpsr.umich.edu/)
Kaggle	Anlaufstelle für die Data-Science-Community, hostet eine Vielzahl an Datensätzen (Kaggle 2022; https://www.kaggle.com/)
Networkrepository	Repositorium mit wissenschaftlichen Netzwerkdaten (Rossi und Ahmed 2022; http://networkrepository.com/)
OSF	Repositorium für Forschungsdaten (Center for Open Science 2022; https://osf.io/)
Beispiele für kuratierte Datensätze und Korpora	
Blog Authorship Corpus	Blog-Posts auf blogger.com, sortiert nach Alter und Geschlecht der Autor:innen (Schler et al. 2005; https://www.kaggle.com/datasets/rtatman/blog-authorship-corpus).
ConvoKit Datasets	Datensätze zur Analyse von Gesprächsverläufen und Dialogen (Chang et al. 2020; https://github.com/CornellNLP/Cornell-Conversational-Analysis-Toolkit)
COVID-19 Weibo Data	Zensierte und unzensierte chinesische Weibo-Nachrichten zu COVID-19 (Fu & Zhu 2020; https://doi.org/10.6084/m9.figshare.12199038)
DeReKo	Deutsches Referenzkorpus, weltweit größte Sammlung von Korpora mit belletristischen, wissenschaftlichen und journalistischen Texten der deutschen Gegenwartssprache (Kupietz et al. 2010; https://www.ids-mannheim.de/digspra/kl/projekte/korpora/).
DiDi-Korpus	Anonymisierte Kommentare und Chat-Nachrichten von Südtiroler Facebook-Nutzer:innen (Frey et al. 2019; https://clarin.eurac.edu/repository/xmlui/handle/20.500.12124/7)

(Fortsetzung)

Tab. 2.2 (Fortsetzung)

Beispiele für kuratierte Datensätze und Korpora	
Dortmunder Chat-Korpus	Chatprotokolle aus professionellen Kontexten und dem Freizeitbereich (Beißwenger 2013; Zugang über https://vlo.clarin.eu/)
D-Place	Database of Places, Language, Culture, and Environment. Ökonomische, soziodemographische und klimatische Bedingungen verschiedener Gesellschaftsformen zur Beschreibung der Menschheitsgeschichte (Jahre 1600–1990) (Kirby et al. 2016; https://d-place.org/)
FiveThirtyEight Russian Troll Tweets	Tweets von Accounts, die mit der Internet Research Agency (IRA) in Verbindung standen. Die russische IRA wird mit Einflussnahmen auf den amerikanischen Wahlkampf im Jahr 2016 in Verbindung gebracht (Linvill und Warren 2018; https://github.com/fivethirtyeight/russian-troll-tweets/)
GDELT	Laufend aktualisierte Datenbank mit weltweiter Berichterstattung. Die Texte durchlaufen eine Vielzahl automatisierter Inhaltsanalysen, zum Beispiel um Ereignisse zu extrahieren (Leetaru 2021; https://gdeltproject.org/)
IMDb	Datenbank zu Filmen und daran beteiligten Personen (Internet Movie Database 2022; https://www.imdb.com/interfaces/)
MoCoDa2	Mobile Communication Database, enthält WhatsApp-Konversationen, die von Nutzer:innen gespendet wurden (Beißwenger et al. 2020; https://db.mocoda2.de/)
Regesta Imperii	Herrscher- und Papsturkunden von den Karolingern bis hin zu Maximilian I. (751–1519) (Akademie der Wissenschaften und der Literatur Mainz 2022; http://www.regesta-imperii.de/)
Seshat	Historische Daten zur sozialen und politischen Organisation menschlicher Gesellschaften bis zur industriellen Revolution (Evolution Institute und Seshat Project 2021; http://seshatdatabank.info/)
Trump Tweets	Alle Tweets von Donald Trump (bis 20. Januar 2020) (Reese 2020; https://www.kaggle.com/austinreese/trump-tweets)
UCDP GED	Weltweite Beobachtung politischer Konflikte mit Angaben zu Todesfällen und beteiligten Akteuren (Uppsala Conflict Data Program 2016; https://ucdp.uu.se/)
useNews	Datensatz zur Verbreitung von Nachrichtenartikeln, enthält eine Kombination von Befragungsdaten, Metadaten von Artikeln und Daten zu Interaktionen auf Social-Media-Plattformen (Puschmann und Haim 2021; https://osf.io/uzca3/)

(Fortsetzung)

Tab. 2.2 (Fortsetzung)

Quellen für Trainingsmaterial	
GermEval SemEval	Veranstaltungen, bei denen im Rahmen von Aufgaben Datensätze zur automatischen Klassifikation bereitgestellt werden, etwa zu Offensive Language oder Sentiment-Analyse (GermEval 2019; https://germeval.github.io/; SemEval 2022; https://semeval.github.io/)
Oxford Text Archive	Literarische und linguistisch aufbereitete Texte, zum Beispiel zur Entwicklung automatischer Textanalyse (University of Oxford 2021; https://ota.bodleian.ox.ac.uk/repository/xmlui/)
TIGER Korpus	Textkorpus mit etwa 900.000 annotierten Token aus deutschen Nachrichtenartikeln (Brants et al. 2004; https://www.ims.uni-stuttgart.de/forschung/ressourcen/korpora/tiger/).
Twitter Event Datasets	Sammlung von 30 unterschiedlichen Twitter-Datensätzen (2012–2016) zur Überprüfung der Replizierbarkeit von Twitter-Studien (Zubiaga 2018a; 2018b; https://figshare.com/articles/Twitter_event_datasets_2012-2016_/5100460)
UCI Machine Learning Repository	Trainingsdaten und Referenzdatensätze unter anderem für die Bild- und Texterkennung (Dua & Graff 2019; https://archive.ics.uci.edu/https://archive.ics.uci.edu/ml/datasets.php)
VoxCeleb	Ton- und Videoaufnahmen zum Beispiel für die Entwicklung von Sprach-, Gesichts- und Emotionserkennung (Visual Geometry Group 2022; http://www.robots.ox.ac.uk/~vgg/data/voxceleb/)
Wikipedia: List of datasets for machine-learning research	Auflistung von Trainingsdatensätzen für das Machine Learning, unter anderem zur Bilderkennung (Gesichts-Objekt-, Handschrifterkennung) und Texterkennung (Rezensionen, Nachrichten, Dialoge etc.) (Wikipedia 2022a; https://en.wikipedia.org/wiki/List_of_datasets_for_machine-learning_research)

Quelle: Eigene Darstellung. Beachten Sie, dass die Datenqualität sehr unterschiedlich ausfällt, und vor der Verwendung der Daten eingeschätzt werden sollte

möglichen (Sekundärdatenanalyse). Eine zentrale Anlaufstelle für sozialwissenschaftliche Daten ist in Deutschland die GESIS. Mit dem Rückenwind der Open-Access-Bewegung fordern auch internationale Zeitschriften von ihren Autor:innen immer häufiger, dass die Daten verfügbar gemacht werden. Eine globale Plattform dafür betreibt die Open Science Foundation,[10] aber auch einzelne Universitäten bieten eigene Repositorien an. Zudem wird in Deutschland momentan mit viel Auf-

[10] Auch Sie selbst können Ihre Daten und Auswertungsskripte dort verfügbar machen. Wenn Sie selbst im Kontext von Qualifikationsarbeiten (B.A., M.A., Promotion) forschen, denken Sie darüber nach!

wand eine nationale Forschungsdateninfrastrukur aufgebaut,[11] über die Forschungsdaten nachhaltig nutzbar gemacht werden sollen. Die Anforderungen an solche wissenschaftlichen Datenbestände werden mit den FAIR-Prinzipien beschrieben – Findability, Accessibility, Interoperability und Reusability (Wilkinson et al. 2016).

Kuratierte Datensätze zu spezifischen Themen gehen teilweise auf wissenschaftliche Forschungsprojekte zurück oder werden von wissenschaftsnahen Organisationen für die Grundlagenforschung erstellt. Hierzu zählen beispielsweise von CLARIN bereit gestellte Sprachdaten, die von D-Place zusammengetragenen Strukturdaten zu menschlichen Gesellschaftsformen oder die vom UCDP erfassten Daten zu politischen Konflikten. Im Social-Media-Bereich sammelt beispielsweise Weiboscope Datensätze, mit denen sich die chinesische Zensur untersuchen lässt (Weiboscope 2022; zum Beispiel zu COVID-19, siehe Fu und Zhu 2020). Im Rahmen geisteswissenschaftlicher Grundlagenforschung werden insbesondere historische Daten in Langzeitprojekten erschlossen, die häufig an den Akademien der Wissenschaften angesiedelt sind. Diese Projekte verweisen auf eine lange Tradition, so hat die Aufarbeitung von königlichen und päpstlichen Urkunden des Mittelalters im Projekt Regesta Imperii bereits im Jahr 1829 begonnen (Akademie der Wissenschaften und der Literatur Mainz 2022). Regelmäßig werden in geisteswissenschaftlichen Projekten gedruckte Editionen veröffentlicht und die erschlossenen Daten werden zunehmend online in Datenbanken zur Verfügung gestellt.

Diese Datenbanken sind im Wesentlichen an zwei unterschiedlichen Zielstellungen ausgerichtet. Erstens erlauben einige Datenbestände inhaltliche Analysen, etwa um das Twitter-Verhalten von Donald Trump oder die Debatten im deutschen Bundestag auszuwerten. Zweitens stellen einige Projekte Daten als **Trainingsmaterial** für Machine-Learning-Verfahren bereit. Hier geht es darum, automatisierte Inhaltsanalysen zu entwickeln. Dazu gehören auch Korpora mit Videos für die Entwicklung automatischer Emotionserkennung oder Korpora mit Rezensionen für die Erkennung positiver und negativer Bewertungen. Ergänzend zu den inhaltlichen Daten sind diese Korpora manuell annotiert, das bedeutet zu jedem Video ist eine zusätzliche Angabe der Emotion oder zu jeder Rezension eine von Menschen vorgenommene Bewertung vorhanden. In jedem Fall sollten Sie sich genau mit der Qualität der Daten beschäftigen und darauf achten, dass deren Entstehung und Auswahl nachvollziehbar sind. So wie Sie keine Texte zitieren sollten, in denen die Aussagen nicht belegt oder begründet sind, sind auch undokumentierte Daten für wissenschaftliche Analysen ungeeignet.

Zum Erstellen annotierter Daten greifen einige Wissenschaftler:innen und Unternehmen auf **Crowdsourcing** zurück. Besonders bekannt, aber auch umstritten, ist zur Rekrutierung von Kodierer:innen für einfache Aufgaben die Plattform Ama-

[11] Siehe NFDI (2022; https://www.nfdi.de).

zon Mechanical Turk.[12] Annotierte Datensätze werden mitunter auf Veranstaltungen zum gemeinsamen Lernen (Hackathon oder Datathon genannt) oder bei Wettbewerben in Kooperation mit Unternehmen ausgegeben oder erstellt. Auf Plattformen wie Kaggle werden laufend Wettbewerbe zur Analyse von Datensätzen ausgeschrieben. Für die Analyse privater Kommunikation, etwa WhatsApp-Konversationen, sind Wissenschaftler:innen auf Datenspenden angewiesen (siehe zum Beispiel Beißwenger et al. 2020; Araujo et al. 2021).

Zusammenfassend unterscheiden sich die verschiedenen Datenzugänge also erstens danach, ob sie heterogene Datenbestände sammeln und durchsuchbar machen oder ob sie sich auf einzelne Themenbereiche beschränken. Zweitens werden Daten speziell für wissenschaftliche Zwecke erzeugt oder treten als Nebenprodukt von Handlungen auf. Die Verbreitung von informationstechnischen Systemen bringt es mit sich, dass umfangreiche Daten über menschliches Verhalten anfallen, mit denen alte, aber auch ganz neue Fragestellungen bearbeitet werden können. Drittens sind einige Datensätze nicht in erster Linie für inhaltliche Fragestellungen ausgelegt, sondern als Trainingsmaterial für die Methodenentwicklung. Viertens werden Datensätze nicht nur von wissenschaftlichen Einrichtungen mit entsprechenden Qualitätssicherungsverfahren, sondern auch von kommerziellen Anbietern bereit gestellt. Da letztere an marktwirtschaftlichen Prinzipien orientiert sind, kann es zu Interessenskonflikten kommen – die Datenqualität sollte vor der Verwendung besonders gründlich eingeschätzt werden. In Tab. 2.2 finden Sie einige Beispiele für die verschiedenen Arten von Datenquellen – verschaffen Sie sich selbst einen Eindruck davon, wie diese Daten einzuschätzen sind und begeben Sie sich gegebenenfalls auf die Suche nach weiteren Datensätzen!

Übungsfragen

1. Was unterscheidet Webscraping von der Datenerhebung über APIs?
2. Aus welchen Bestandteilen besteht eine URL?
3. Besuchen Sie eine Webseite und prüfen Sie, ob eine API bereit gestellt wird!
4. Schauen Sie sich die Dokumentation des Endpunkts „users“ der Twitter-API an. Suchen Sie dort die Bestandteile der verwendeten URL heraus: Wie lauten Domain, Pfad und Parameter?
5. Was versteht man unter Open Data?
6. Suchen Sie einen im Internet zum Download angebotenen Datensatz und schätzen Sie die Qualität der Daten ein. Was spricht gegen die wissenschaftliche Verwendung der gefundenen Daten, was spricht dafür?
7. Worin unterscheiden sich die Datenbanken von wissenschaftlichen Repositorien und die Datenbanken von Social-Media-Plattformen?

[12] Siehe Amazon Mechanical Turk (2018; https://www.mturk.com/).

Weiterführende Literatur

Fielding, R. T. (2000). *Architectural styles and the design of network-based software architectures.* Dissertation, University of California.

Jünger, J. (2018). Mapping the field of automated data collection on the web. Data types, collection approaches and their research logic. In C. M. Stützer, M. Welker & M. Egger (Hrsg.), *Computational social science in the age of big data. Concepts, methodologies, tools, and applications* (S. 104–130). Köln: Halem.

Russell, M. A. (2014). *Mining the social web. Data mining Facebook, Twitter, Linkedin, Google+, GitHub, and more.* (2. Aufl.). Sebastopol: O'Reilly.

Literatur

Abgeordnetenwatch.de. (2022). *Abgeordnetenwatch API Dokumentation*. Zugriff am 30.05.2022. https://www.abgeordnetenwatch.de/api

Akademie der Wissenschaften und der Literatur Mainz. (2022). *Regesta Imperii*. Zugriff am 30.05.2022. http://www.regesta-imperii.de/

Amazon Mechanical Turk. (2018). *Amazon Mechanical Turk. Access a global, on-demand, 24x7 workforce*. Zugriff am 23.05.2022. https://www.mturk.com/

Amazon Web Services. (2022a). *Amazon Rekognition. Automatisieren Sie Ihre Bild- und Videoanalyse mit Machine Learning*. Zugriff am 30.05.2022. https://aws.amazon.com/de/rekognition/

Araujo, T., Ausloos, J., van Atteveldt, W., Loecherbach, F., Moeller, J., Ohme, J. et al. (2021). *OSD2F: An Open-Source Data Donation Framework*. https://doi.org/10.31235/osf.io/xjk6t

BDZV. (2022). *Bundesverband Digitalpublisher und Zeitungsverleger*. Zugriff am 18.05.2022. https://www.bdzv.de/

Beißwenger, M. (2013). Das Dortmunder Chat-Korpus. *Zeitschrift für germanistische Linguistik*, *41*(1), 161–164. https://doi.org/10.1515/zgl-2013-0009

Beißwenger, M., Fladrich, M., Imo, W. & Ziegler, E. (2020). Die Mobile Communication Database 2 (MoCoDa 2). In K. Marx, H. Lobin & A. Schmidt (Hrsg.), *Deutsch in Sozialen Medien* (S. 349–352). Berlin: de Gruyter. https://doi.org/10.1515/9783110679885-018

Berlin-Brandenburgische Akademie der Wissenschaften. (2022). *API (Schnittstellen zum DWDS)*. Zugriff am 30.05.2022. https://www.dwds.de/d/api

Berlind, D. (2022). *ProgrammableWeb*. Zugriff am 18.05.2022. https://www.programmableweb.com/

Brants, S., Dipper, S., Eisenberg, P., Hansen, S., König, E., Lezius, W. et al. (2004). TIGER: Linguistic Interpretation of a German Corpus. *Journal of Language and Computation*, *2004*(2), 597–620. https://www.ims.uni-stuttgart.de/forschung/ressourcen/korpora/tiger/

Bruns, A. (2019). After the 'APIcalypse': social media platforms and their fight against critical scholarly research. *Information, Communication & Society*, *22*(11), 1544–1566. https://doi.org/10.1080/1369118X.2019.1637447

Center for Open Science. (2022). *Open Science Foundation*. Zugriff am 18.05.2022. https://osf.io/

Chang, J. P., Chiam, C., Fu, L., Wang, A., Zhang, J. & Danescu-Niculescu-Mizil, C. (2020). *ConvoKit: A Toolkit for the Analysis of Conversations*. Proceedings of SIGDIAL. Zugriff am 30.05.2022. https://github.com/CornellNLP/ConvoKit

CKCEST: China Knowledge Centre for Engineering Sciences and Technology. (2022). *AMiner. Datasets for Social Network Analysis*. Zugriff am 30.05.2022. https://cn.aminer.org/data-sna

CLARIN. (2022). *CLARIN Virtual Language Observatory*. Zugriff am 21.06.2022. https://vlo.clarin.eu/

Datasette. (2022). *Openregister. Custom SQL query*. Zugriff am 30.05.2022. https://db.offeneregister.de/openregister

Destatis. (2022). *Statistisches Bundesamt*. Zugriff am 16.05.2022. https://www.destatis.de/

Dua, D. & Graff, C. (2019). *UCI Machine Learning Repository,* University of California, School of Information and Computer Science. Zugriff am 30.05.2022. https://archive.ics.uci.edu

Entity Reconciliation Community Group. (2022). *Reconciliation Service API v0.1. A protocol for data matching on the Web*. Zugriff am 16.05.2022. https://reconciliation-api.github.io/specs/0.1/

Evolution Institute & Seshat Project. (2021). *Seshat: Global History Databank*. Zugriff am 30.05.2022. http://seshatdatabank.info/

Fandom. (2022). *Fandom. Search the world's largest fan wiki platform*. Zugriff am 16.05.2022. https://www.fandom.com/

Fielding, R. (2000). *Architectural styles and the design of network-based software architectures*. Dissertation. Irvine: University of California.

Fiesler, C., Beard, N. & Keegan, B. C. (2020). No Robots, Spiders, or Scrapers: Legal and Ethical Regulation of Data Collection Methods in Social Media Terms of Service. *Proceedings of the International AAAI Conference on Web and Social Media*, *14*(1), 187–196. https://ojs.aaai.org/index.php/ICWSM/article/view/7290

Frey, J.-C. & Glaznieks, Aivars and Stemle, Egon W. (2019). *DIDI – The DiDi Corpus of South Tyrolean CMC 1.0.0,* Eurac Research CLARIN Centre. Zugriff am 30.05.2022. https://clarin.eurac.edu/repository/xmlui/handle/20.500.12124/7

Fu, K. & Zhu, Y. (2020). Did the world overlook the media's early warning of COVID-19? *Journal of Risk Research*, *23*(7–8), 1047–1051. https://doi.org/10.1080/13669877.2020.1756380

GermEval. (2019). *GermEval Shared Task Hub. Natural Language Processing shared tasks for German*. Zugriff am 30.05.2022. https://germeval.github.io/

Google. (2014). *Public Data*. Zugriff am 30.05.2022. https://www.google.com/publicdata/directory

Google. (2022d). *Cloud Vision API. Vision AI*. Zugriff am 30.05.2022. https://cloud.google.com/vision

Google. (2022f). *Dataset Search*. Zugriff am 30.05.2022. https://datasetsearch.research.google.com/

Google Developers. (2022). *YouTube. Let users watch, find, and manage YouTube content*. Zugriff am 30.05.2022. https://developers.google.com/youtube/

GovData. (2022). *Das Datenportal für Deutschland. Open Government: Verwaltungsdaten transparent, offen und frei nutzbar*. Zugriff am 30.05.2022. https://www.govdata.de/
Harvard University. (2022a). *Harvard Dataverse*. Zugriff am 18.05.2022. https://dataverse.harvard.edu/
Harvard University. (2022b). *Social Science One,* Harvard's Institute for Quantitative Social Science. Zugriff am 23.05.2022. https://socialscience.one/
Huynh, D. (2022). OpenRefine (Version 3.5.2) [Computer software]: Metaweb Technologies. https://openrefine.org/
Hydra W3C Community Group. (2018). *API Documentation*. Zugriff am 18.05.2021. http://www.hydra-cg.com/drafts/use-cases/2.api-documentation.md
IBM. (2022). *IBM Watson Studio. Erstellen und skalieren Sie vertrauenswürdige KI in jeder Cloud. Automatisieren Sie den KI-Lebenszyklus für ModelOps.* Zugriff am 30.05.2022. https://www.ibm.com/de-de/cloud/watson-studio
International Movie Database. (2022). *IMDb Datasets*. Zugriff am 23.05.2022. https://www.imdb.com/interfaces/
IVW. (2022). *Informationsgesellschaft zur Feststellung der Verbreitung von Werbeträgern*. Zugriff am 16.05.2022. https://www.ivw.de/
Jacobson, D., Brail, G. & Woods, D. (2012). *APIs: A strategy guide. Creating channels with application programming interfaces.* Beijing: O'Reilly.
Johnson, B. & Turner, L. A. (2003). Data Collection Strategies in Mixed Methods Research. In A. Tashakkori & C. Teddlie (Hrsg.), *Handbook of Mixed Methods in Social & Behavioral Research* (S. 297–319). Thousand Oaks: Sage.
Jünger, J. (2021). A brief history of APIs. How social media providers shape the opportunities and limitations of online research. In U. Engel, A. Quan-Haase, S. X. Liu & L. Lyberg (Hrsg.), *Handbook of Computational Social Science* (S. 17–32). London: Routledge. https://doi.org/10.4324/9781003025245-3
Kaggle. (2022). *Kaggle: Your Machine Learning and Data Science Community*. Zugriff am 23.05.2022. https://www.kaggle.com/
Kirby, K. R., Gray, R. D., Greenhill, S. J., Jordan, F. M., Gomes-Ng, S., Bibiko, H.-J. et al. (2016). D-PLACE: A Global Database of Cultural, Linguistic and Environmental Diversity. *PloS one*, *11*(7), e0158391. https://doi.org/10.1371/journal.pone.0158391
Kotsios, A., Magnani, M., Vega, D., Rossi, L. & Shklovski, I. (2019). An Analysis of the Consequences of the General Data Protection Regulation on Social Network Research. *ACM Transactions on Social Computing*, *2*(3), 1–22. https://doi.org/10.1145/3365524
Kupietz, M., Belica, C., Keibel, H. & Witt, A. (2010). The German Reference Corpus DeReKo: A Primordial Sample for Linguistic Research. *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC'10).* 1848–1854. http://www.lrec-conf.org/proceedings/lrec2010/pdf/414_Paper.pdf
Leetaru, K. H. (2021). *The GDELT Project*. Zugriff am 08.05.2022. https://www.gdeltproject.org/
GESIS. (2022). *GESIS-Suche*. Zugriff am 18.05.2022. https://search.gesis.org/
Linvill, D. & Warren, P. (2018). *3 million Russian troll tweets*. Zugriff am 30.05.2022. https://github.com/fivethirtyeight/russian-troll-tweets/
Media Cloud (2022). *Media Cloud is an open-source platform for media analysis*. Zugriff am 30.05.2022. https://mediacloud.org/
MediaWiki. (2022, 16. Mai). *API:Main page*. Zugriff am 30.05.2022. https://www.mediawiki.org/wiki/API:Main_page

Meta. (2021). *CrowdTangle. A tool from Meta to help follow, analyze, and report on what's happening across social media.* Zugriff am 23.05.2022. https://www.crowdtangle.com/

Meta. (2022). *Meta for Developers*. Zugriff am 30.05.2022. https://developers.facebook.com/

Microsoft. (2022a). *Azure Cognitive Services documentation. Learn how to use ready-made AI services to build intelligent apps, websites, and bots. Develop software that can see, hear, speak, and interpret your user's needs.* Zugriff am 30.05.2022. https://docs.microsoft.com/en-us/azure/cognitive-services/

Ministerium der Justiz Nordrhein-Westfalen. (2020). *Gemeinsames Registerportal der Länder*. Zugriff am 16.05.2022. https://www.handelsregister.de/rp_web/welcome.xhtml

NFDI. (2022). *Nationale Forschungsdaten Infrastruktur*. Zugriff am 15.01.2023. https://www.nfdi.de

OpenAPI Initiative (2022). *The OpenAPI Specification Repository*. Zugriff am 16.05.2022. https://github.com/OAI/OpenAPI-Specification

OpenCorporates. (2022). *Get data access to over 200 million companies*. Zugriff am 30.05.2022. https://api.opencorporates.com/

OpenRefine. (2021). *Reconcilable Data Sources*. Zugriff am 18.05.2022. https://github.com/OpenRefine/OpenRefine/wiki/Reconcilable-Data-Sources

Peroni, S. & Daquino, M. (2020). *The unifying REST API for all the OpenCitations Indexes*. Zugriff am 30.05.2022. https://opencitations.net/index/api/v1

Perspective. (2021). *Using machine learning to reduce toxicity online. Perspective API can help mitigate toxicity and ensure healthy dialogue online*. Jigsaw. Zugriff am 30.05.2022. https://www.perspectiveapi.com/

Puschmann, C. & Haim, M. (2021). *useNews*. [Dataset]. Zugriff am 28.01.2022. https://osf.io/uzca3/

RatSWD. (2019). *Big Data in den Sozial-, Verhaltens- und Wirtschaftswissenschaften: Datenzugang und Forschungsdatenmanagement.* Berlin: RatSWD. https://doi.org/10.17620/02671.39

Reese, A. (2020). *Trump Tweets. Tweets from @realdonaldtrump*. [Dataset], kaggle.com. Zugriff am 30.05.2022. https://www.kaggle.com/datasets/austinreese/trump-tweets

Rossi, L. & Ahmed, N. K. (2022). *Network Repository. A Scientific Network Data Repository with Interactive Visualization and Mining Tools*. Zugriff am 30.05.2022. https://networkrepository.com/

Schler, J., Koppel, M., Argamon, S. & Pennebaker, J. (2005). Effects of Age and Gender on Blogging. *Proceedings of 2006 AAAI Spring Symposium on Computational Approaches for Analyzing Weblogs*. http://www.cs.biu.ac.il/~schlerj/schler_springsymp06.pdf

SemEval. (2022). *International Workshop on Semantic Evaluation*. Zugriff am 30.05.2022. https://semeval.github.io/

Thelwall, M. [Mike] & Stuart, D. (2006). Web crawling ethics revisited: Cost, privacy, and denial of service. *Journal of the American Society for Information Science and Technology*, *57*(13), 1771–1779. https://doi.org/10.1002/asi.20388

Twitter. (2022d). *Twitter Developer Platform. Use Cases, Tutorials, & Documentation*. Zugriff am 30.05.2022. https://developer.twitter.com/en

University of Michigan. (2022). *ICPSR: Inter-University Consortium for Political and Social Research*. Zugriff am 30.05.2022. https://www.icpsr.umich.edu/web/pages/

University of Oxford. (2021). *OTA: Oxford Text Archive. A repository of full-text literary and linguistic resources. Thousands of texts in more than 25 languages*. Zugriff am 30.05.2022. https://ota.bodleian.ox.ac.uk/repository/xmlui/

Uppsala Conflict Data Program. (2016). *Georeferenced Event Dataset (GED). Global version 17.1*. [Dataset]. http://ucdp.uu.se/downloads/

Visual Geometry Group. (2022). *VoxCeleb. A large scale audio-visual dataset of human speech*. Zugriff am 30.05.2022. https://www.robots.ox.ac.uk/~vgg/data/voxceleb/

Weiboscope. (2022). *HKU JMSC Weibo Censorship Index*. Journalism and Media Studies Centre. Zugriff am 23.05.2022. https://weiboscope.jmsc.hku.hk/wsr/

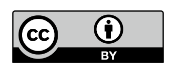
Wikimedia Deutschland. (2022). *Wikipedia. Die freie Enzyklopädie*. Zugriff am 16.05.2022. https://www.wikipedia.de/

Wikipedia. (2022a). *List of datasets for machine-learning research*. https://en.wikipedia.org/wiki/List_of_datasets_for_machine-learning_research

Wilkinson, M. D., Dumontier, M., Aalbersberg, I. J. J., Appleton, G., Axton, M., Baak, A. et al. (2016). The FAIR Guiding Principles for scientific data management and stewardship. *Scientific Data*, *3*, 1–9. https://doi.org/10.1038/sdata.2016.18

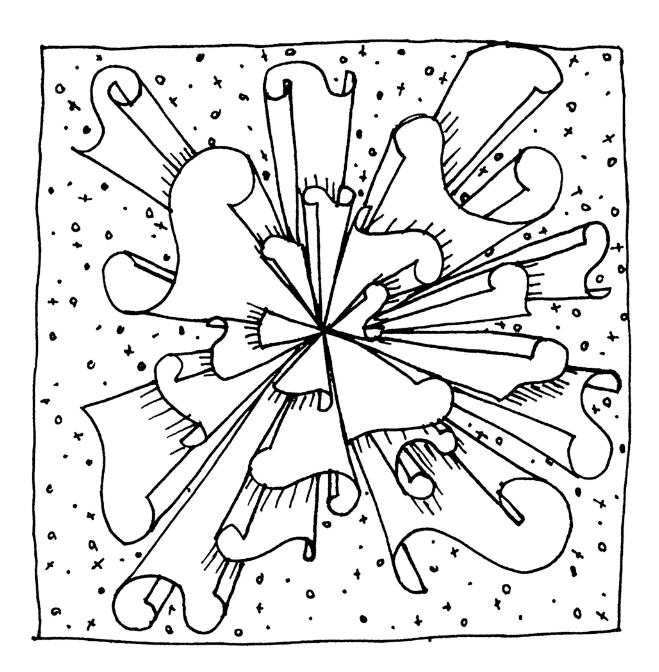
Zubiaga, A. (2018a). A longitudinal assessment of the persistence of twitter datasets. *Journal of the Association for Information Science and Technology*, *69*(8), 974–984. https://doi.org/10.1002/asi.24026

Zubiaga, A. (2018b). *Twitter event datasets 2012–2016*. [Dataset]. Zugriff am 30.05.2022. https://doi.org/10.6084/m9.figshare.5100460.v2





3 Datenformate



J. Jünger, C. Gärtner, *Computational Methods für die Sozial- und Geisteswissenschaften*, https://doi.org/10.1007/978-3-658-37747-2_3

Zusammenfassung

Das Kapitel führt in Datenformate ein, die Ihnen bei der Arbeit mit Computational Methods begegnen werden. Sie lernen, wie die wichtigsten Datenformate aufgebaut sind, welche Arten von Datenbanken es gibt und worin der Unterschied zwischen Datenformaten und Datenmodellen besteht.

Im Online-Repositorium unter https://github.com/strohne/cm finden Sie begleitend zum Kapitel weitere Materialien, auf die wir im Text mit ☛ verweisen.

Schlüsselwörter

Datentypen · Markdown · Zeichenkodierung · CSV · HTML und XML · JSON · SQL und NoSQL · Datenmodell · Resource Description Framework (RDF) · Semantic Web

Das Wort ‚Daten' ist eines der meistverwendeten Wörter in diesem Buch. Etymologisch leitet sich die Bezeichnung von dem lateinischen ‚datum' ab, was als ‚gegeben' übersetzt werden kann (Kluge 2002, S. 181). Als Datum werden in der ursprünglichen Bedeutung die Zeit- und Ortsangaben auf Schriftstücken bezeichnet (Kluge 2002, S. 181). Diese Verwendungsweise findet sich auch heute noch, die Extension des Begriffs, das heißt der Umfang als Datum bezeichneter Gegenstände, hat sich aber stark ausgeweitet. Je nach Perspektive werden darunter etwa Zahlen, Beobachtungen oder Bits verstanden (Ballsun-Stanton 2010). Vor allem das letzte Verständnis, Bits als Kodierung von Information, findet sich im Kontext der Informatik wieder. In diesem Sinne ist der Begriff sogar in einer ISO-Norm[1] standardisiert und wird dort definiert als „A reinterpretable representation of information in a formalized manner suitable for communication, interpretation, or processing" (ISO/IEC 2382-1, siehe ISO 2015). Ein wichtiges Merkmal ist dabei, dass es sich um Repräsentationen von etwas handelt. Daten stehen zum Beispiel für das Verhalten von Menschen oder allgemeiner für Informationen.[2] Dabei kann

[1] Die International Organization for Standardization (ISO) erarbeitet internationale Normen für viele Bereiche, unter anderem informationstechnische Normen.

[2] Die Verbindung von Daten, Informationen und Wissen wird häufig als Pyramide modelliert, wobei von einem zunehmenden Bedeutungsgehalt ausgegangen wird. Aus Sicht der Informatik bestehen Analysen daraus, dass Informationen aus Daten extrahiert werden, auf denen dann handlungsleitendes Wissen basiert. Diese Vorstellung ist jedoch zu hinterfragen, da Informationen aus sozialwissenschaftlicher Sicht nicht in Daten enthalten sein können, sondern lediglich zugeschrieben werden. Zur DIKW-Pyramide (data, information, knowledge, wisdom) siehe Ackoff (1989) und Rowley (2007).

es verschiedene Repräsentationen für die gleichen Dinge geben, das heißt, Daten können unterschiedlich angeordnet bzw. formatiert sein (☛ *Repositorium*). In diesem Kapitel werden zunächst verschiedene Datentypen und dann verbreitete Datenformate vorgestellt. Für eine Repräsentation in Tabellenform eignet sich das CSV-Format. Sollen große Datenbestände verwaltet werden, kommen häufig SQL-Datenbanken zum Einsatz, in denen mehrere Tabellen enthalten sein können. Im Prinzip kann fast jede Datenstruktur in Tabellenform gebracht werden, das ist aber nicht immer gut handhabbar. Eine höhere Flexibilität in Bezug auf die Strukturierung bieten HTML und XML für Textdaten oder JSON für Daten, die aus einzelnen Werten zusammengesetzt sind. Auch solche flexiblen Datenbestände werden bei entsprechendem Umfang in Datenbankmanagementsystemen vorgehalten, die dann als NoSQL-Datenbanken bezeichnet werden.

Die genannten Datenformate geben kaum Vorgaben darüber, welche Bedeutung die Daten haben. Die Bedeutung wird erst mit Datenmodellen festgelegt, in denen die Verwendung bestimmter Datenelemente inhaltlich definiert ist. So kann man definieren, dass zu einer Personenbeschreibung ein Name, ein Geburtsdatum und ein Wohnort gehören. Insbesondere im World Wide Web besteht der Bedarf, solche Merkmale zu standardisieren, um die Datenbestände verschiedener Anbieter verknüpfen zu können. Die Idee wird Semantic Web genannt und das zugrundeliegende Datenmodell nennt sich Ressource Description Framework (RDF). Dieses Modell – und verschiedene dazugehörige Datenformate – werden am Ende des Kapitels vorgestellt.

3.1 Datentypen

Daten sind in ihrer kleinsten Einheit Zeichen, die von Computern verarbeitet werden. Diese Zeichen haben verschiedene Datentypen, sie sind zum Beispiel Zahlen (integer, double, float), Zeichenketten (string, character) oder Wahrheitswerte (boolean). Menschen können meistens auf den ersten Blick erkennen, um welchen Datentyp es sich handelt, Computern muss das hingegen erst gesagt werden. So könnte ein Programm eine 13 sowohl als die ganze Zahl 13, als die Dezimalzahl 13,0 oder als eine Abfolge von einzelnen Buchstaben „1“ und „3“ erkennen. Diese Unterscheidung ist wichtig, da je nach Datentyp unterschiedliche Operationen möglich sind. So kann man mit Zahlen beispielsweise rechnen. Zeichenketten wie das Wort „Datum“ kann man zwar nicht multiplizieren oder dividieren, stattdessen könnte man in dem Wort aber beispielsweise nach Vokalen suchen.

Einzelne Werte mit solchen grundlegenden, atomaren Datentypen werden mathematisch als **Skalare** bezeichnet. Atomare Datentypen können wiederum zu komplexeren Datentypen zusammengesetzt werden. Eine Liste gleichartiger Elemente wird **Vektor** genannt, zum Beispiel eine Liste mit Datumsangaben. Fasst man mehrere solcher Vektoren zusammen, entsteht eine **Matrix** (Abb. 3.1). Für viele datenanalytische Fragestellungen eignen sich Matrizen, wenn etwa jede Zeile für eine Person steht und in den Spalten die Eigenschaften der Personen festgehalten sind. Eine Matrix setzt sich also aus einer Liste mit Vektoren zusammen, je nach Perspektive aus einer Liste mit Zeilen oder aus einer Liste mit Spalten.

Matrizen im engeren Sinn enthalten nur einen einzigen Datentyp, zum Beispiel nur Jahreszahlen (Ganzzahl) oder nur Geburtsorte (Zeichenkette). Um mehrere Datentypen gleichzeitig zu erfassen, kommen in verschiedenen Programmiersprachen noch eine ganze Reihe weiterer komplexer Datentypen vor. Sehr flexibel sind Diktionäre, die Name-Wert-Paare in Listen erfassen und tief verschachtelt sein können, das heißt ein Wert kann wiederum ein Diktionär sein. Für die Datenanalyse wird zudem meist mit Tabellen gearbeitet, die ähnlich wie eine Matrix aus Zeilen und Spalten bestehen, in den verschiedenen Spalten aber ganz unterschiedliche Datentypen enthalten können. Beispiele für atomare und komplexe Datentypen in den Programmiersprachen R und Python finden sich in Tab. 3.1.

Immer wenn etwas nicht wie erwartet funktioniert, empfiehlt es sich, die Datentypen zu überprüfen. Besonders wichtig ist es, bei der Programmierung und Datenanalyse zu unterscheiden, wann es sich um Zeichenketten (mit Anführungszei-

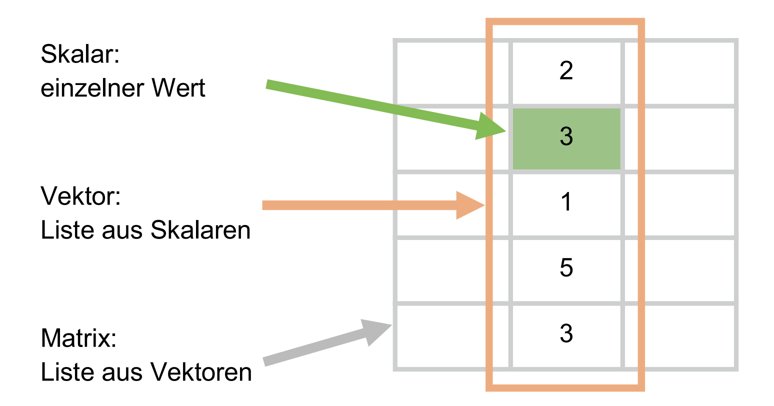


Abb. 3.1 Aufbau von Matrizen. (Quelle: eigene Darstellung)

chen), Zahlen (ohne Anführungszeichen) oder festgelegte Bezeichnungen bzw. Eigennamen von Datenobjekten (ohne Anführungszeichen) handelt. Dabei können nicht beliebige Anführungszeichen eingesetzt werden, sondern es ist in jeder Sprache festgelegt, wann einfache oder doppelte Zeichen verwendet werden müssen. Insbesondere typografische Anführungszeichen „“ wie sie in Textprogrammen automatisch bei der Eingabe entstehen – in Deutschland erst Anführungszeichen unten, dann oben – funktionieren nicht, darauf sollte beim Kopieren von Daten und Quelltexten geachtet werden.

Tab. 3.1 Beispiele für atomare und zusammengesetzte Datentypen

Datentyp	**Beschreibung**	**Beispiel** und *Bezeichnung* in R	**Beispiel** und *Bezeichnung* in Python
Atomare Datentypen			
Zahlen	Ganze Zahlen (numeric)	1884L *integer*	1884 *int*
Dezimal-zahlen	Zahlen mit Nachkommastellen (numeric)	1.6 *double*	1.6 *float*
Wahrheits-werte	Logische Werte (boolean)	TRUE, FALSE *logical*	TRUE, FALSE *bool*
Zeichen-ketten	Ketten von Buchstaben, Zahlen oder Sonderzeichen (string)	"Anna" *character*	"Anna" *str*
Zusammengesetzte Datentypen			
Vektor	Liste von Elementen des gleichen Typs	c(1,2,3)	np.array([1,2,3])
Liste	Ansammlung von Elementen mit unterschiedlichen Typen	list('Anna',1884)	['Anna',1884]
Tupel	Zusammensetzung von Elementen mit unterschiedlichen Typen	tuple('Anna',25)	('Anna',25)
Diktionär	Name-Wert-Paare (dict, dictionary)	list('Name'='Anna', 'Jahr'=1884)	{'Name': 'Anna', 'Jahr': 1884}
Matrix	Liste von Vektoren bzw. zweidimensionales Array	matrix(1:4, nrow=2)	np.array([1,2],[3,4])
Tabelle	Datensatz in Tabellenform, bestehend aus Zeilen, Spalten und Zellen	*data.frame* *tibble*	*pd.DataFrame*

Quelle: Eigene Darstellung

3.2 Textformate (MD)

Viele der im Folgenden besprochenen Datenformate sind einfache Textdateien, die mit einem Editor wie Atom oder Notepad++ geöffnet und bearbeitet werden können (siehe Kap. 1). So wie verschiedene natürliche Sprachen – Deutsch, Englisch, Farsi – durch eine Syntax, Vokabulare und eine bestimmte Aussprache geprägt sind, legen die Datenformate jeweils fest, welche Zeichen wie kodiert werden. Damit ein Computerprogramm die Dateien auch versteht, sind über alle Datenformate hinweg zwei Besonderheiten zu beachten.

Erstens sind nicht alle Zeichen sichtbar. Als **Zeilenumbruch** werden in Textdateien unter Windows zwei unsichtbare Zeichen verwendet, das Carriage Return (CR; Wagenrücklauf) gefolgt vom Line Feed (LF). Diese Zeichen sind Steuerzeichen und simulieren eine Schreibmaschine, bei der am Ende einer Zeile zunächst das Papier nach rechts geschoben wurde, sodass sich die Schreibposition am Zeilenanfang befindet. Dann wurde die Walze mit dem Papier zu einer neuen Zeile weitergedreht.

Mitunter werden Zeilenumbrüche durch ihre ASCII[3]-Werte `#13` für Carriage Return und `#10` für Line Feed dargestellt, manchmal auch durch die Escape-Sequenzen `\r` für Carriage Return und `\n` für Line Feed. In Unix-Systemen wie macOS oder Linux wird ausschließlich der Line Feed verwendet. In älteren Mac-OS-Varianten war dagegen das Carriage Return gebräuchlich. Die verschiedenen Möglichkeiten sind eine häufige Fehlerquelle beim Einlesen von Datensätzen. Im Zweifelsfall empfiehlt es sich, alle Zeilenumbrüche zunächst mit einem Texteditor durch ein einzelnes Line-Feed-Zeichen zu ersetzen. Um in einem Texteditor zu sehen, welche Zeichen in einer Datei als Zeilenumbruch fungieren, können Sie zum einen die Steuerzeichen einblenden, häufig steht dafür eine mit dem Absatzsymbol ¶ gekennzeichnete Schaltfläche zur Verfügung. Die Zeilenumbrüche sollten nun am Ende jeder Zeile sichtbar markiert sein. Zum anderen wird in Texteditoren üblicherweise das Format der Zeilenumbrüche in der Statusleiste aufgeführt und kann mit einem Klick geändert werden.[4]

[3] Der American Standard Code for Information Interchange ist im Jahr 1963 entstanden und umfasst 128 Zeichen, die bis heute die Grundlage für fast alle computerbasierten Zeichensysteme bilden.

[4] In Notepad++ und Atom ist beispielsweise in der unteren Leiste neben der Zeichenkodierung der Zeilenumbruch eingeblendet – „Windows (CR LF)" oder „Macintosh (CR)". Mit einem Links- bzw. Rechtsklick können Sie das Format ändern. Alternativ ist in Atom die Funktion Line Ending Selector: Convert To [...] über die Kommando-Palette (Tastenkombination Strg bzw. Cmd + Shift + P) erreichbar.

Zweitens muss die Kodierung der verschiedenen lokalen und internationalen Zeichen – arabische Schriftzeichen, chinesische Glyphen, deutsche Umlaute, französische Akzente oder auch Emojis – beachtet werden. Als **Zeichenkodierung** (engl. *encoding*) wird, wenn keine Umlaute oder andere Sonderzeichen enthalten sind, normalerweise das ASCII-Format verwendet. Für Umlaute oder andere Sonderzeichen kommen je nach Sprache unterschiedliche Erweiterungen dieses Formats zur Anwendung, die zum Teil mit verwirrenden Bezeichnungen versehen sind. Besonders verbreitet sind ISO-8859-1 (auch ANSI genannt) sowie Windows-1252 (auch Latin-1 oder CP1252 genannt).

Mit ANSI lassen sich bis zu 256 verschiedene Zeichen darstellen. Immer mehr setzt sich allerdings Unicode durch. Dieser Standard ist mit der Idee verbunden, alle Zeichen der Welt zu erfassen, enthält zum Beispiel auch Emojis und fortlaufend werden Anträge auf neue Emojis gestellt (Unicode 2021). Um diese Vielfalt vollständig abzubilden, werden mehrere Bytes benötigt, das entsprechende UTF-16-Format verwendet deshalb zwei Bytes und UTF-32 sogar vier Bytes je Zeichen. Als platzsparende Variante kommt meistens UTF-8 zum Einsatz und das aus gutem Grund: Mit UTF-8 lässt sich nicht nur jedes bislang über Unicode standardisierte Zeichen sehr platzsparend kodieren. Es ist zudem abwärtskompatibel, denn wenn keine Sonderzeichen benötigt werden, dann ist die Kodierung identisch zum ASCII-Format.[5]

Bei UTF-8 wird für die ASCII-Zeichen ein Byte verwendet, für alle weiteren Zeichen dagegen zwei Bytes. Immer wenn zwei Bytes verwendet werden, gibt es wiederum verschiedene Möglichkeiten, welches Byte als erstes angegeben ist. Im Zweifelsfall wird dies durch eine Byte Order Mark (BOM) gekennzeichnet.[6] Dabei handelt es sich um ein Zeichen ganz am Anfang der Textdatei. Einige CSV-Dateien enthalten dieses Zeichen, andere nicht. Werden Sonderzeichen also in einer geöffneten Datei nicht richtig angezeigt – wenn etwa anstelle von „Hüte“ ein „HÃ¼te“ erscheint – lohnt es sich, die Zeichenkodierung zu überprüfen und gegebenenfalls mit einem Texteditor umzustellen.

In Textdateien können beliebige Inhalte festgehalten werden, nicht nur vorstrukturierte Daten, sondern zum Beispiel auch eigene Notizen. Diese Dateien können unkompliziert mit anderen geteilt werden, da nahezu auf jedem Computer

[5] Für ASCII werden sieben Bit verwendet, ein achtes Bit kennzeichnet dann, ob ASCII oder ein erweiterter Zeichensatz wie ANSI oder UTF-8 vorliegt. Deshalb sind alle ASCII-Zeichen im ANSI-Zeichensatz und auch in UTF-8 enthalten.

[6] Bei UTF-16 unterscheidet man deshalb UTF-16LE (little endian) und UTF-16BE (big endian), damit sind die beiden unterschiedliche Reihenfolgen der zwei Bytes gemeint.

mindestens ein Programm zum Bearbeiten von Textdateien vorhanden ist. Allerdings sind darin normalerweise keine Formatierungen wie Fettungen oder Kursivierungen möglich. Um dennoch zumindest grundlegende Formatierungen vornehmen zu können, hat sich in vielen Bereichen das Format Markdown etabliert:

- Überschriften beginnen mit einer Raute:
  ```
  # Überschrift 1
  ## Überschrift 2
  ```
- Aufzählungen werden durch Bindestriche realisiert:
  ```
  - Erster Punkt
  - Zweiter Punkt
  ```
- Text wir kursiv geschrieben, indem er in Sternchen eingeschlossen wird:
  ```
  *kursiver Text*
  ```
- Für fetten Text werden zwei Sternchen verwendet:
  ```
  **fetter Text**
  ```
- Code wird in Backticks eingeschlossen, um darin zum Beispiel Sternchen oder Bindestriche verwenden zu können. Backticks sind rückwärtsgerichtete Anführungszeichen und auf der Tastatur etwas versteckt:
  ```
  `Quellcode im Text`
  ```
- Soll Quellcode über mehrere Zeilen gehen, dann werden zu Beginn und am Ende jeweils drei Backticks gesetzt:
  ````
  ```
  Code über
  mehrere Zeilen.
  ```
  ````

Absätze werden durch eine Leerzeile erzeugt und aufpassen sollte man bei Zeilenumbrüchen. Denn ein einzelner Zeilenumbruch wird nicht immer als Zeilenumbruch dargestellt. Zeilenumbrüche in Aufzählungen funktionieren beispielsweise nur dann, wenn die vorangegangene Zeile mit zwei Leerzeichen endet und der folgende Text mit zwei Leerzeichen eingerückt wird. Darüber hinaus sind viele weitere Formatierungen zur Darstellung von Links, Bildern oder Tabellen möglich. Eine Übersicht bieten Cheatsheets, die in großer Zahl im Internet zu finden sind.[7]

Während einfache Textdateien meistens mit der Endung *.txt* abgespeichert werden, sind Markdown-Dateien an der Endung *.md* zu erkennen. Typischerweise fin-

[7] Siehe zum Beispiel Cone (2022; https://www.markdownguide.org/cheat-sheet/).

det sich beispielsweise in Git-Repositorien immer eine Datei *readme.md*, in der eine Einführung in das Repositorium gegeben wird. Auf der Webseite des Repositoriums werden die Markdown-Formatierungen dann automatisch in HTML umgeformt, sodass im Browser echte Listen, Kursivierungen oder Überschriften erscheinen. Schauen Sie sich einmal den Quelltext der readme-Datei im ☛ Repositorium und die Darstellung auf der Webseite an!

3.3 Tabellenformate (CSV)

Eines der am meisten verbreiteten Formate, um tabellarische Daten zu speichern, ist das CSV-Format. Hierbei handelt es sich um eine Textdatei, in der jede Zeile einen Fall enthält, zum Beispiel einen Kommentar auf einer Online-Plattform oder Angaben zu einer Person in einem Roman. Innerhalb einer Zeile sind die Werte mit einem Komma getrennt. Die Abkürzung des Dateiformats steht dementsprechend für Comma Separated Values. In Abb. 3.2 ist ein Auszug aus einer solchen Datei abgebildet, in der Tweets mit einer ID und zugehörigen Kennwerten aufgelistet sind. Die Spaltennamen sind in der ersten Zeile angegeben. Hier wird auch eine Besonderheit sichtbar: Die Liste der Hashtags ist in der zweiten Zeile in Anführungszeichen gesetzt, damit sie als ein Wert zählt, obwohl darin ebenfalls ein Komma enthalten ist.

Es gibt allerdings etliche Varianten dieses Formats und man muss im Einzelfall prüfen, womit man es zu tun hat. Die Dateien unterscheiden sich zum einen wie alle Textdateien durch die verwendeten Zeilenumbrüche und durch die Zeichenkodierung (siehe oben). Dazu kommen zwei Besonderheiten von CSV-Dateien:

- Als **Trennzeichen** zwischen den Werten sind neben Kommata auch Semikola, Leerzeichen, Tabulator oder seltener die Pipe (senkrechter Strich `|`) oder die Raute (Doppelkreuz `#`) gebräuchlich. Je nach Voreinstellung des Systems kommen Microsoft Office-Programme meistens besser mit Tabulatoren zurecht, andere Statistikprogramme vor allem mit Kommata und Semikola.

```
id,name,from,favorites,replies,retweets,hashtags
6,eaduenergy,Forschungslabor Eadu,64,0,1,"sternzerstörer,werft"
7,eaduenergy,Forschungslabor Eadu,3,0,,todesstern
8,eaduenergy,Forschungslabor Eadu,30,0,6,kyber
```

Abb. 3.2 Auszug aus einer CSV-Datei. (Quelle: eigene Darstellung)

Soll in einem Feld das Trennzeichen selbst verwendet werden, dann muss dieses markiert werden, damit dadurch keine neue Spalte beginnt. Häufig werden in diesem Fall die Werte in doppelte Anführungszeichen gesetzt. Dadurch verschiebt sich das Problem, denn auch Anführungszeichen können in den Werten selbst vorkommen. Dieses Problem wird wiederum anders gelöst und als Maskieren bezeichnet: Entweder wird den Anführungszeichen ein Backslash vorangestellt oder die Anführungszeichen werden verdoppelt. Soll wiederum ein solches Maskierungszeichen wie der Backslash verwendet werden, wird es nochmals maskiert, das heißt verdoppelt. Diese Form des Maskierens von reservierten Zeichen wird in vielen anderen Sprachen wie R oder Python ähnlich umgesetzt.

- Die **Spaltenbezeichnungen** sind häufig in der ersten Zeile angegeben, aber nicht immer. Dann müssen die Spaltennamen beim Einlesen zusätzlich angegeben werden.

Sollten Sie CSV-Dateien in Programme einlesen, um diese weiterzuverarbeiten, können Konvertierungsprobleme auftreten. Dies passiert, wenn das Programm von einem anderen CSV-Dialekt ausgeht, als ihn die Datei aufweist. In diesem Fall versuchen Sie zunächst herauszufinden, ob Sie Einstellungen beim Einlesen des Datensatzes vornehmen können, etwa um das Trennzeichen festzulegen. Um einschätzen zu können, welcher ‚Dialekt' in einer CSV-Datei eingesetzt wird, kann man sie mit einem Texteditor öffnen (siehe Kap. 1; ☛ *Repositorium*). Wichtig ist dabei, dass man im Texteditor auch die unsichtbaren Steuerzeichen einblendet. Sonst lassen sich Tabulatoren und Leerzeichen oder auch die verschiedenen Zeilenumbruchszeichen nicht voneinander unterscheiden. Mit einem Texteditor können Sie das Format der Datei ggf. ändern, indem Sie beispielsweise alle Tabulatoren durch Semikola ersetzen. Auch bei anderen Dateiformaten lohnt es sich in der Regel, sie einmal mit einem Texteditor zu betrachten.

3.4 Auszeichnungssprachen (HTML und XML)

Um in einem fortlaufenden Text einzelne Abschnitte zu markieren und zu formatieren, eignen sich Auszeichnungssprachen. Häufig verwendete Auszeichnungssprachen sind die Hypertext Markup Language (HTML) oder die Extensible Markup Language (XML). Die Grundprinzipien von Auszeichnungssprachen sind meist ähnlich. So geht es zunächst darum, allgemein die gesamte Struktur sowie einzelne Merkmale in Texten zu annotieren. Umgesetzt wird dies durch das Umklammern einzelner Textteile.

Im Detail unterscheiden sich die verschiedenen Auszeichnungssprachen in ihrer syntaktischen Umsetzung und in den Anwendungsfällen. HTML ist die Standard-

sprache, mit der Webseiten geschrieben werden. Soll beispielsweise ein Wort fett dargestellt werden, so wird es mit einem `<b>`-Tag umschlossen. Der Name des Tags ist in spitzen Klammern angegeben, der Buchstabe „b“ steht in diesem Fall für „bold“ (deutsch *fett*). Direkt vor der Textstelle wird ein öffnendes Tag gesetzt und direkt danach ein schließendes Tag. Der Unterschied zwischen diesen beiden Varianten besteht darin, dass schließende Tags nach der ersten spitzen Klammer einen Schrägstrich / enthalten:

```
Dieser Satz enthält ein <b>fett</b> gedrucktes Wort.
```

Nur öffnende Tags können mit weiteren Attributen versehen werden. Solche Attribute werden abgetrennt mit einem Leerzeichen hinter dem Namen des Tags angegeben. Auf den Namen des Attributs folgt ein Gleichheitszeichen, danach in einfachen oder doppelten Anführungszeichen der Wert. Attribute sind also Name-Wert-Paare. So kann ein Tag mit einem eindeutigen Bezeichner (ID) versehen werden, damit es von anderen gleichnamigen Tags unterscheidbar ist. Häufig sind auch `class`-Attribute anzutreffen, die für die Gestaltung im Web eine besondere Rolle spielen (siehe Abschn. 4.1).

```
Dieser Satz enthält zwei <b id="wort1" class="gruen">fett
</b> gedruckte Wörter mit <b id="wort2" class="gruen">
unterschiedlichen</b> IDs und der gleichen Klasse.
```

Durch Tags markierte Bereiche können auch verschachtelt werden, etwa um einen Absatz zu markieren und darin Wörter hervorzuheben. Absätze werden durch `<p>`-Tags markiert. Allerdings dürfen sich die Bereiche nicht überkreuzen. Wenn also eine Fettung im nächsten Absatz fortgesetzt werden soll, muss sie zunächst geschlossen und dann nach dem öffnenden Absatz-Tag wieder neu geöffnet werden:

```
<p>Dieser Absatz endet mit <b>fett gedruckten Wörtern
</b></p><p><b>Dieser Absatz</b> beginnt mit fett
gedruckten Wörtern.</p>
```

Die Möglichkeit zur hierarchischen Strukturierung ist eine besondere Stärke von Auszeichnungssprachen und macht sie flexibel einsetzbar. Außerdem erlaubt der HTML5-Standard in bestimmten Fällen auch alleinstehende Tags, die genauso aussehen wie öffnende Tags und nicht wieder geschlossen werden müssen. Damit wird der Quelltext übersichtlicher, etwa wenn Bilder eingebunden werden:

```
<img src="chewbacca.png" alt="Chewbacca">
```

Sprachen wie HTML legen darüber hinaus auch den Aufbau von Dokumenten fest. Ein HTML5-Dokument beginnt immer mit der Angabe des Dokumententyps und dem `<html>`-Element. Darin sind ein `<head>`-Element für nicht sichtbare allgemeine Angaben und ein `<body>`-Element mit den sichtbaren Inhalten der Seite enthalten. Das Grundgerüst sieht wie folgt aus:

```
<!DOCTYPE html>
<html>
  <head>
     <title>Seitentitel</title>
  </head>
  <body>
     Seiteninhalte
  </body>
</html>
```

Auch das Vokabular ist standardisiert, das heißt die Tags haben eine festgelegte Bedeutung. So sind im HTML5-Standard unter anderem folgende Tags festgelegt:[8]

- `<h1>`, `<h2>` bis `<h6>` kennzeichnen Überschriften erster, zweiter bis sechster Ordnung.
- `<p>` kennzeichnet Absätze.
- `<em>` kennzeichnet Hervorhebungen.
- `<table>` kennzeichnet eine Tabelle, `<tr>` eine Tabellenzeile und `<td>` eine Zelle innerhalb einer Tabellenzeile.
- `<img>` wird für Bilder verwendet.
- `<a>` steht für Anchor und wird zum Setzen von Links verwendet.
- `<meta>` steht für Metanagaben wie das Erstelldatum oder der Titel einer Seite. Diese Angaben sind im Kopfbereich des Quelltextes eingebettet und nicht sichtbar.

Für die Formatierung von Webseiten sind noch weitere Elemente verbreitet, die keine vorgegebene Bedeutung haben. In Verbindung mit IDs und Klassen-Attributen werden diese Elemente dann mittels Cascading Style Sheets (CSS) optisch formatiert oder mittels JavaScript um interaktive Funktionen erweitert:

[8] Der HTML-Standard wird durch das W3C festgelegt und ist dort auch dokumentiert, siehe W3Schools (2022a; https://www.w3schools.com/tags/default.asp).

- `<span>` kennzeichnet Textbereiche, in der Regel kurze Phrasen oder einzelne Wörter.
- `<div>` kennzeichnet Blöcke, zum Beispiel den Kopfbereich oder den Fußbereich einer Seite.

Um einen Eindruck davon zu gewinnen, wie eine Webseite aufgebaut ist, können Sie sich den Quelltext im Browser ansehen. Hierzu kann in den meisten Browsern mit der rechten Maustaste ein Kontextmenü aufgerufen werden, in dem die Funktion SEITENQUELLTEXT ANZEIGEN oder ähnlich angeboten wird. Der gesamte Quelltext ist häufig jedoch eher unübersichtlich. Zudem interessiert in der Regel nur ein ganz bestimmter Ausschnitt aus dem Quelltext. Um schnell zu erfassen, wie ein bestimmter Teil einer Seite umgesetzt wurde, lässt sich eine Entwicklerkonsole öffnen. In der Konsole kann gezielt ein Ausschnitt aus dem Quelltext anvisiert werden. Dazu klicken Sie im Browser mit der rechten Maustaste auf ein Element, zum Beispiel ein Eingabefeld, und wählen dann ELEMENT UNTERSUCHEN (Abb. 3.3). Alternativ können Sie die Entwicklerkonsole in vielen Browsern mit

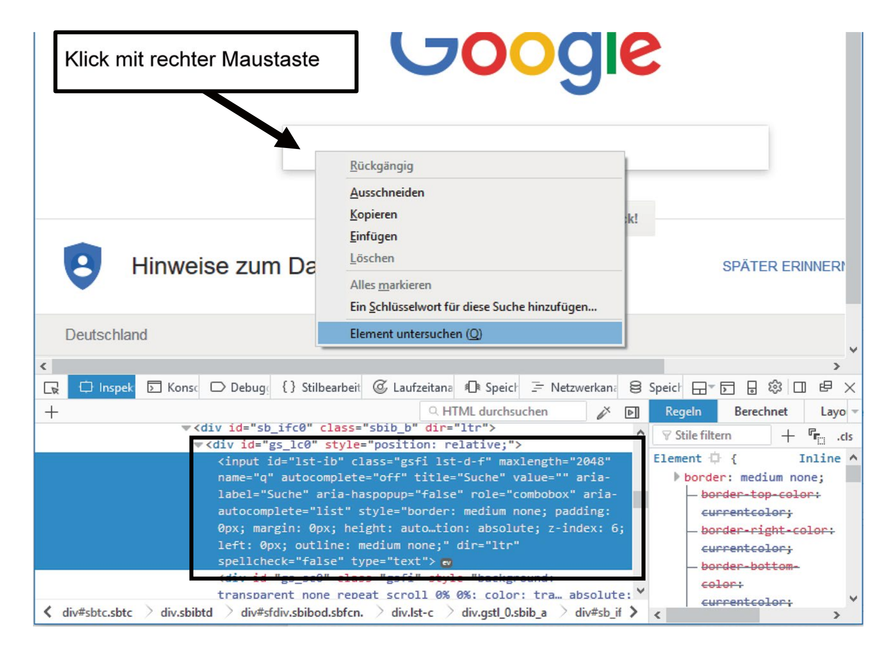


Abb. 3.3 Der Quelltext von Webseiten im Browser Firefox (Entwicklerkonsole). (Quelle: eigene Darstellung)

der Taste F12 erreichen. Sie können dort auch Texte oder Attribute verändern, probieren Sie es einmal aus!

Der Browser arbeitet nicht direkt mit dem Quelltext, sondern liest ihn in ein sogenanntes Document Object Model (DOM) ein. Dieser Vorgang heißt parsen und spielt im Zusammenhang mit automatisierter Datenerhebung eine Rolle (siehe Kap. 7). Das DOM bildet den Quelltext in einer Baumstruktur bestehend aus Knoten ab. Die Tags, Attribute und der Text werden dazu in Knoten umgewandelt. Dieses DOM bzw. die enthaltenen Knoten können durch Sprachen wie JavaScript verändert werden, um eine Webseite interaktiv zu gestalten.

Eine weitere Auszeichnungssprache, die vor allem als Datenbankformat eingesetzt wird, stellt die Extensible Markup Language (XML) dar.[9] Im Grunde funktioniert XML ähnlich wie HTML, auch mithilfe dieser Sprache werden Texte strukturiert. Im Gegensatz zu HTML ist XML allerdings in der inhaltlichen Ausgestaltung flexibler, was speziell an bestimmte Anwendungsfälle angepasste Datenstrukturen erlaubt. Die Flexibilität ist vor allem dadurch begründet, dass in XML fast beliebige Namen für Tags und Attribute verwendet werden können. Hier ist hauptsächlich die Syntax festgelegt, das heißt die Struktur aus Tags, Attributen und Text. Außerdem unterscheidet sich XML insofern von HTML, dass in XML-Dokumenten alle Tags immer geschlossen werden. Ein Tag kann bei Bedarf geöffnet und sofort wieder geschlossen werden, indem ein Schrägstrich vor die schließende spitze Klammer gestellt wird:

```
<img src="chewbacca.png" alt="Chewbacca" />
```

Viele dieser Auszeichnungssprachen liegen in unterschiedlichen Versionen vor. Während XML nur die allgemeine Struktur vorgibt, ist die Bedeutung der Elemente in weitergehenden Standards festgelegt. Zum Beispiel baut das TEI-Format,[10] ein in den Geisteswissenschaften verbreiteter Standard zur Aufbereitung von Texten, auf XML auf. Im TEI-Format ist beispielsweise festgelegt, dass Sätze oder Nebensätze mit bestimmten Tags wie `<li>` oder `<cl>` versehen werden, sodass die Satzstruktur rekonstruiert werden kann. Auch RSS-Feeds, in denen zum Beispiel Nachrichtenseiten die aktuellen Meldungen anbieten, sind ein XML-Format. Ebenso enthalten einige Microsoft-Office-Dateien eine Sammlung von XML-Dokumenten mit einer festgelegten Struktur (OpenDocument, Office Open XML),

[9] Auch der XML-Standard ist von W3C dokumentiert und einsehbar unter W3C (2013; https://www.w3.org/TR/xml/).

[10] Siehe Text Encoding Initiative (2022; https://tei-c.org/).

die zu einer Datei zusammengepackt und komprimiert sind.[11] Auch für die Annotation von Trainingskorpora für das Machine Learning (siehe Kap. 8) wird XML eingesetzt. Im Oxford Text Archive (siehe Kap. 2) sind zum Beispiel Personen in Dokumenten als sogenannte Named Entities gekennzeichnet.

3.5 Objektdatenformate (JSON)

Innerhalb von Programmiersprachen werden Daten häufig als Objekte repräsentiert, die Eigenschaften mit bestimmten Werten haben. Eine Nutzerin könnte dann beispielsweise als Objekt erfasst sein, welches die Eigenschaft „Name" mit dem Wert „Eliza" hat. Werte können auch komplexe Datentypen umfassen – insbesondere Listen, wenn jemand mehrere Namen hat – oder die Werte selbst sind wiederum Objekte, wenn ein Namensobjekt zum Beispiel die Eigenschaften „Vorname" und „Nachname" enthält.

Mit derartigen Datenstrukturen lassen sich viele Bereiche der Welt innerhalb von Programmen modellieren. Dafür werden Listen, Name-Wert-Paare und elementare Datentypen wie Zahlen, Zeichenketten und Datumstypen benötigt. Ein Datenformat für solche Datenstrukturen ist JSON (JavaScript Object Notation).[12] Dieses Format wird im Web – dem Ursprungskontext der Programmiersprache JavaScript – zum Beispiel für APIs (Programmierschnittstellen) oder für interaktiv nachgeladene Inhalte verwendet.[13] Mittlerweile hat sich JSON in vielen weiteren Bereichen etabliert. So lassen sich zum Beispiel mit der Programmiersprache Python (siehe Abschn. 5.2) unter Verwendung der *json*-Bibliothek die internen Datenstrukturen als JSON abspeichern oder laden.[14]

In JSON werden Listen in eckige Klammern eingefasst, wobei die Elemente mit Kommata getrennt werden. Eine Sammlung von Name-Wert-Paaren wird als Dictionary bezeichnet und ebenfalls mit Kommata getrennt, aber in geschweiften Klammern zusammengefasst. Der Name (auch Schlüssel, engl. *key*) wird links von

[11] Zum Auspacken einer docx-Datei kann ein normales ZIP-Programm wie 7zip verwendet werden. Anschließend lassen sich die Inhalte im Texteditor ansehen. Probieren Sie es einmal aus!

[12] Eine Referenz zum JSON-Format wird von der Internet Engineering Task Force (IETF) herausgegeben, siehe Bray (2014; https://tools.ietf.org/html/rfc7159).

[13] Die zum interaktiven Nachladen verwendeten XMLHttpRequests vermitteln entgegen dem Namen nicht nur XML, sondern auch andere Datenformate wie JSON.

[14] Achtung bei der Verwendung von Anführungszeichen: JSON sieht ähnlich aus wie Dictionaries in Python (Tab. 3.1), allerdings dürfen hier anders als in Python nur doppelte Anführungszeichen verwendet werden.

einem Doppelpunkt angegeben, der Wert rechts davon. Zahlenwerte werden angegeben wie sie sind. Die Schlüssel und Zeichenketten werden in doppelte Anführungszeichen gesetzt:

```
[
    {
      "Name": "Eliza",
      "Typ" : "Bot",
      "Geburtsjahr" : 1966
    },
    {
      "Name": {
        "vorname": "Joseph",
        "nachname": "Weizenbaum"
      },
      "Typ": "Mensch",
      "Geburtsjahr": 1923
    }
]
```

Ein Vorteil gegenüber proprietären[15] Formaten zur Repräsentation von Objekten besteht darin, dass JSON sowohl maschinen- als auch menschenlesbar ist. Wie auch CSV-Dateien und HTML- oder XML-Dateien können JSON-Dateien mit einem Texteditor geöffnet und bearbeitet werden. Die Lesbarkeit ist vor allem dann gut, wenn die Hierarchie der Struktur durch Einrückungen sichtbar gemacht wird. Das gilt natürlich auch für XML und HTML. Einige Texteditoren stellen Funktionen bereit, um sinnvoll einzurücken. Dieser Vorgang und die entsprechenden Funktionen nennen sich im Englischen *pretty print*.

3.6 Datenbanken (SQL und NoSQL)

Jede Sammlung von CSV-, HTML- oder JSON-Dateien kann man bereits als Datenbank im weiteren Sinn begreifen. Wenn von einer Datenbank die Rede ist, meint man damit aber meistens ein Datenbank Management System (DBMS). Ein sol-

[15] Der Ausdruck „proprietär“ bedeutet, dass ein Hersteller ein Datenformat ohne Rücksicht auf Standards und häufig auch ohne öffentliche Dokumentation speziell für seine eigenen Produkte entwickelt hat. Proprietäre Software ist vereinfacht gesagt das Gegenstück zu Open-Source-Software.

ches System ist mehr als die Zusammenstellung von Daten. Dazu gehört in der Regel eine Software, die Funktionen zur effizienten Verwaltung (Speichern, Abfragen, Lastverteilung) und auch zum Sicherstellen der Datensicherheit (Transaktionen, Verschlüsselung) bereitstellt. Diese Funktionen sind einerseits für den Umgang mit großen Datenbeständen und andererseits für die Koordination gleichzeitiger Zugriffe nötig.

Im Allgemeinen unterscheidet man relationale und nichtrelationale Datenbanken. Tendenziell eignen sich relationale Datenbanken immer dann, wenn das Schema der erfassten Daten festgelegt ist und tabellarische Daten schnell aus der Datenbank ausgelesen werden sollen. Nichtrelationale Datenbanken sind dahingegen besonders dann von Vorteil, wenn vermehrt veränderliche Datenstrukturen oder Objekte in die Datenbank geschrieben werden. Vom Datenbankmanagementsystem hängt auch ab, über welche Sprache Daten in die Datenbank geschrieben oder aus dieser gelesen werden. Besonders bei relationalen Datenbanken kommt die Abfragesprache SQL (Structured Query Language) zum Einsatz. Nichtrelationale Datenbanken werden auch NoSQL-Datenbanken genannt, beispielsweise werden Netzwerke in Graphendatenbanken erfasst und über SPARQL oder die Cypher Query Language abgefragt (siehe Kap. 4).

Relationale Datenbanken legen die Daten in Tabellen ab, die untereinander über IDs verknüpft sind (engl. *join*). In einer Tabelle sind dann beispielsweise Blogartikel abgelegt und in einer anderen die Kommentare der Nutzer:innen zu einem Artikel. Jeder Artikel und jeder Kommentar erhalten eine ID, das heißt eine eindeutige Nummer.[16] In der Tabelle mit den Kommentaren wird zusätzlich die ID des dazugehörigen Artikels als sogenannter Fremdschlüssel abgelegt (siehe auch Abb. 3.4).

Man unterscheidet dabei vor allem drei Arten von Beziehungen:

- Bei 1:1-Beziehungen gehört zu einem Datensatz immer genau ein Datensatz aus einer anderen Tabelle. Zum Beispiel wird einer Person immer genau eine Postadresse zugeordnet.
- Bei 1:n-Beziehungen oder n:1-Beziehungen können zu einem Datensatz mehrere Datensätze aus einer anderen Tabelle gehören. Ein:e Verkäufer:in kann zum Beispiel mehrere Produkte verkaufen. Bei der Umsetzung erhält jede:r Verkäufer:in eine eindeutige ID (= Schlüssel) und diese ID wird in einer Spalte der Produkttabelle (= Fremdschlüssel) vermerkt.

[16] Es können theoretisch auch nichtnumerische Merkmale verwendet werden (z. B. Hashwerte). Die Verwendung von Nummern ist allerdings platzsparend und effizient, da Computer Zahlen gut verarbeiten können.

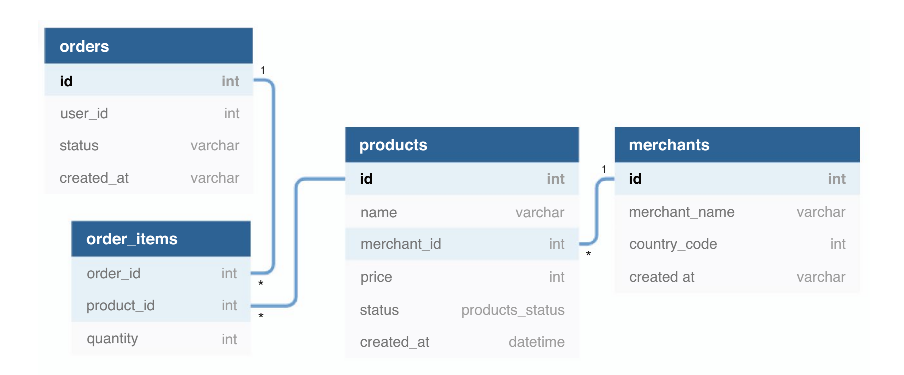


Abb. 3.4 Auszug aus der relationalen Datenstruktur eines Shop-Systems. Jeder Kasten steht für eine Tabelle, darin sind die Spalten aufgeführt. Die Verknüpfung erfolgt über IDs. Zum Beispiel wird in der Tabelle für die Elemente im Warenkorb (order_items) die ID des Produkts vermerkt. (Quelle: Holistics.io (2022; https://dbdiagram.io/))

- Bei n:m-Beziehungen werden mehreren Datensätzen aus einer Tabelle mehrere Datensätze aus einer anderen Tabelle zugeordnet. Zum Beispiel kann eine Bestellung mehrere Produkte umfassen, gleichzeitig kann ein Produkt mehrfach von unterschiedlichen Käufer:innen bestellt werden. Diese Struktur wird meistens über drei Tabellen abgebildet: eine Produkttabelle, eine Bestelltabelle und eine Warenkorbtabelle. Erst indem die Warenkorbtabelle zu den anderen beiden Tabellen in 1:n-Beziehungen steht, stehen die Produkt- und Bestelltabelle zueinander in einer n:m-Beziehung.

Die Struktur der Tabellen – vor allem Name der Tabellen, Name und Datentyp der Spalten – ist in der Regel festgelegt. Diese Struktur kann zwar verändert werden, betrifft dann aber den gesamten Datenbestand. Eine relationale Datenbank muss also den damit abgebildeten Gegenstand (z. B. Weblogs oder Online-Shops) vollständig modellieren.

Eines der am weitesten verbreiteten Datenbankmanagementsysteme ist MariaDB. Es ist als Open-Source-Software verfügbar, eine kommerzielle Lizenzierung erlaubt das kompatible MySQL.[17] Für den Datenaustausch verwendet man bei SQL-Datenbanken häufig sogenannte Dumps. Dabei handelt es sich um Textdateien, in denen die SQL-Befehle zum Erzeugen der Datenbank, der Tabellen und

[17] MariaDB (MariaDB Foundation 2022; https://mariadb.org/) ist in eine Abspaltung von MySQL (Oracle 2021; https://www.mysql.com/), beide Varianten sind meist austauschbar.

```
-- Struktur der Tabelle pages
CREATE TABLE IF NOT EXISTS `pages` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `created` datetime DEFAULT CURRENT_TIMESTAMP,
  `title` text COLLATE utf8_unicode_ci,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;

-- Daten der Tabelle pages
INSERT INTO `pages` (`id`, `created`, `title`) VALUES
(7, '2018-09-08 13:59:05', 'Kontakt'),
(8, '2018-09-08 13:59:05', 'Startseite'),
(9, '2018-09-08 13:59:05', 'Impressum')
```

Abb. 3.5 Auszug aus dem SQL-Dump eines Content-Management-Systems für Webseiten. (Quelle: eigene Darstellung)

der Inhalte aufgelistet sind (Abb. 3.5). Die Befehle können dann in einem Datenbankmanagementsystem ausgeführt werden, um eine Datenbank einzuspielen (siehe Abschn. 4.1.4). Da diese Dateien sehr groß werden können, sind sie meistens mit einem zip-Programm gepackt.

Dagegen ist bei nichtrelationalen Datenbanken die Struktur bzw. das Datenbankschema üblicherweise nicht festgelegt und wird durch die konkreten Datensätze bestimmt. Diese Systeme sind meistens für spezielle Einsatzzwecke optimiert. Für einen sehr schnellen und einfachen Datenzugriff können etwa Name-Werte-Listen verwendet werden. Beispiele dafür sind Google Bigtable[18] oder Redis.[19] Da die die Namen der Werte flexibel handhabbar sind und nicht von vornherein festgelegt werden müssen, können später immer wieder neue Datensorten aufgenommen werden. Das ist beispielsweise hilfreich, wenn sich erst später herausstellt, dass für eine:n Nutzer:in nicht nur der Link zum Twitter-Profil, sondern auch zum Instagram-Profil erfasst werden soll.

Weisen die Datensätze noch komplexere Strukturen auf, eignen sich dokumentenorientierte Datenbanken wie MongoDB,[20] in welcher Daten intern als JSON abgelegt werden, oder eXist[21] für XML-Dokumente. Ähnliche Systeme kommen für Volltextsuchmaschinen zum Einsatz, zum Beispiel bei Elasticsearch.[22] Gra-

[18] Siehe Google (2022a; https://cloud.google.com/bigtable).

[19] Siehe Redis (2022; https://redis.io/).

[20] Siehe MongoDB (2022; https://mongodb.com/).

[21] Siehe Meier (2003; http://www.exist-db.org/).

[22] Siehe Elastic (2022; https://elastic.co/elasticsearch/).

phenorientierten Systeme wie Neo4j verwalten wiederum besonders gut Netzwerkdaten.[23] Relationale und nichtrelationale Konzepte lassen sich auch mischen, indem flexible Datenformate wie JSON oder XML in den Spalten einer relationalen Datenbank abgelegt werden. Eine Übersicht über die aktuell verbreiteten Datenbanksysteme und Erläuterungen zum Aufbau finden Sie bei DB-Engines.[24]

3.7 Datenmodelle (RDF)

In Bezug auf Daten*formate* haben sich Standards wie CSV, JSON oder XML und HTML herausgebildet. Auch bei den Daten*banken* genießen einzelne Systeme wie MySQL hohe Popularität. Dennoch ist das Daten*modell* von Anwendung zu Anwendung sehr verschieden. Formal versteht man unter einem Datenmodell die Abbildung der Prozesse und Sachverhalte eines Anwendungsbereichs (auch Domäne genannt) auf die Datenstrukturen. Ein Datenmodell umfasst somit beispielsweise die Bezeichnungen der Datenfelder und die Beziehungen zwischen Tabellen. Datenmodelle sind zunächst abstrakte, konzeptionelle Vorstellungen, wie bestimmte Sachverhalte mithilfe von Daten modelliert werden sollen, beispielsweise das Zusammenspiel von Nutzer:innen, Posts und Kommentaren auf einer Social-Networking-Site oder die Struktur einer historisch-kritischen Edition. Die konkrete Ausgestaltung des Modells geschieht dann über die Erfassung der Daten, die mit festgelegten Datenformaten in einer Datenbank abgelegt werden (Abb. 3.6). In der Regel wird für jede Anwendung – sei es eine Smartphone-App oder ein wissenschaftliches Datenanalyseprojekt – ein eigenes Datenmodell entwickelt. Zur Dokumentation von Datenmodellen werden Sprachen wie die Unified Modeling Language (UML) eingesetzt.[25]

Besonders bei der Arbeit mit großen Datenbeständen oder wenn Daten veröffentlicht werden sollen, gewinnt ein einheitliches Datenmodell an Bedeutung. Ziel ist es dabei, Daten nach einem festgelegten System zu erfassen und abzuspeichern, damit diese auch von anderen verwendet werden können. Das lässt sich anhand von Personendatensätzen verdeutlichen: Das Geburtsdatum einer Person kann unter unterschiedlichen Bezeichnungen wie `Geburtstag`, `Geburtsdatum`, `birthday` oder `birthdate` erfasst sein. Für Menschen ist leicht zu erkennen, dass diese Daten den gleichen Sachverhalt repräsentieren. Computer können die Bedeutung nicht erkennen, sodass sich verschiedene Datensätze nicht ohne Weiteres verbinden lassen.

[23] Siehe Neo4J (2022; https://neo4j.com/).
[24] Siehe Solid IT (2022; https://db-engines.com/).
[25] Siehe OMG (2022; https://uml.org/).

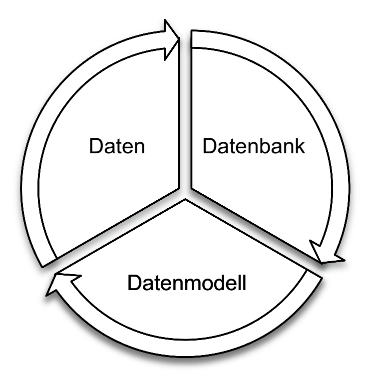


Abb. 3.6 Zusammenhang zwischen Daten, Datenbanken und Datenmodellen. (Quelle: eigene Darstellung)

Ein Lösungsansatz für diese Herausforderung stellt das Resource Description Framework (RDF) dar (siehe W3C 2014). Es umfasst drei wesentliche Konzepte:

1. **Aussagen** setzen sich aus drei Elementen zusammen: einem Subjekt, einem Prädikat und einem Objekt. Ein Geburtsdatum lässt sich wie folgt formulieren: Ada (Subjekt) ist geboren (Prädikat) am 10. Dezember 1815 (Objekt).[26] Im Prinzip lassen sich alle Daten als solche Tripel formalisieren.[27] Mehrere Aussagen lassen sich als Netzwerke bzw. Graphen auffassen. An einem Datum sind zum Beispiel mehrere Personen geboren, sodass verschiedene Personen über das Datum zumindest formal in Beziehung zueinanderstehen.
2. Die Elemente eines Tripels können über **Vokabulare** eindeutig bezeichnet werden. Vokabulare beinhalten festgelegte Ausdrücke für bestimmte Kategorien wie Geburtstage. Ein verbreitetes Vokabular zur Erfassung von Personen ist etwa Friends of a Friend (FOAF), in welchem Prädikate wie Namen, Adressen oder Bekanntschaftsbeziehungen standardisiert sind. Ein Geburtsdatum würde hier mit dem Schlüssel `foaf:birthday` erfasst werden. Weitere Vokabulare werden von schema.org oder DBPedia gepflegt, wobei ein Geburtsdatum über die Schlüssel `schema:birthDate` bzw. `dbo:birthDate` angegeben wird.[28]

---

[26] Diese Struktur stimmt nicht zwangsläufig mit den grammatischen Kategorien der Linguistik überein.

[27] Die Umformung von Datensätzen in diese Struktur kann über das Umformen vom Wide- in das Long-Format erreicht werden (siehe auch Kap. 4).

[28] Siehe Brickley und Miller (2014; http://xmlns.com/foaf/spec), Google et al. (2022; https://schema.org) und DBPedia Association (2021; http://dbpedia.org).

Eine Zusammenstellung untereinander verbundener Begriffe, beispielsweise dass Personen über ein Geburtsdatum und ein Sterbedatum verfügen, wird Ontologie genannt, wobei das Begriffssystem mit Sprachen wie der Web Ontology Language (OWL) formalisiert wird.[29]

3. Bezeichnungen werden entweder in der Form von Zeichenketten (= Literale) oder als eindeutige **Uniform Resource Identifier (URI)**[30] angegeben. Die Präfixe `foaf`, `schema` und `dbo` sind lediglich Abkürzungen, der vollständige URI für `foaf:birthday` lautet beispielsweise `http://xmlns.com/foaf/0.1/birthday`. Selbst wenn verschiedene Dienste für die gleichen Bedeutungen nicht die gleichen Bezeichnungen verwenden, können die Vokabulare ineinander übersetzt werden.[31]

Das Resource Description Framework bringt sehr unterschiedliche Datenmodelle auf den kleinsten gemeinsamen Nenner. Aus dieser Kleinteiligkeit ergibt sich der Nachteil, dass die Daten kaum noch durch Menschen erfasst werden können. Das ist aber auch nicht Ziel dieser Technologie. Vielmehr geht es darum, Daten in maschinenlesbarer Form mit Bedeutung zu versehen und daraus neue Aussagen abzuleiten – sodass nicht Menschen, sondern Maschinen die Daten effizient verknüpfen oder lesen können. Die Daten können dazu in Triple Stores – eine Form nichtrelationaler Datenbanken – abgelegt und über Abfragesprachen wie SPARQL abgefragt werden. Dabei werden Aussagen mit Platzhaltern und Bedingungen formuliert. Eine Abfrage kann zum Beispiel festlegen, dass die Subjekte vom Typ Wissenschaftler:in sein sollen: `?person rdf:type dbo:Scientist`. Mit einer Kombination verschiedener Tripel lässt sich erfragen, welche Wissenschaftler:innen heute Geburtstag haben (Abb. 3.7).

[29] Eine Visualisierung von Begriffsnetzwerken bietet beispielsweise WebVOWL (Link et al. 2019; http://vowl.visualdataweb.org/webvowl.html) und über das W3C Wiki (W3C 2021; https://www.w3.org/wiki/Search_engines) können entsprechende Vokabulare gefunden werden. Die Basic Formal Ontology (BFO) stellt darüber hinaus ein abstraktes Begriffssystem zur Verfügung, das möglichst universell einsetzbar ist (Ruttenberg 2020; http://basic-formal-ontology.org/).

[30] Eine spezielle Form von URIs sind Uniform Resource Locators (URLs, siehe Abschn. 2.1), das heißt Webadressen wie https://de.wikipedia.org/wiki/Ada_Lovelace. Hiermit wird eine Webseite eindeutig identifiziert. Durch die Angabe „https://" ist zudem klar, dass auf diese Resource über das Web (https-Protokoll) zugegriffen werden kann. International Resource Identifiers (IRIs) sind dagegen eine Erweiterung von URIs um Sonderzeichen und Zeichen anderer Sprachen.

[31] Die Vokabulare selbst werden über Metasprachen wie die Web Ontology Language (OWL) oder RDF Schema (RDFS) beschrieben und lassen sich darüber untereinander verbinden.

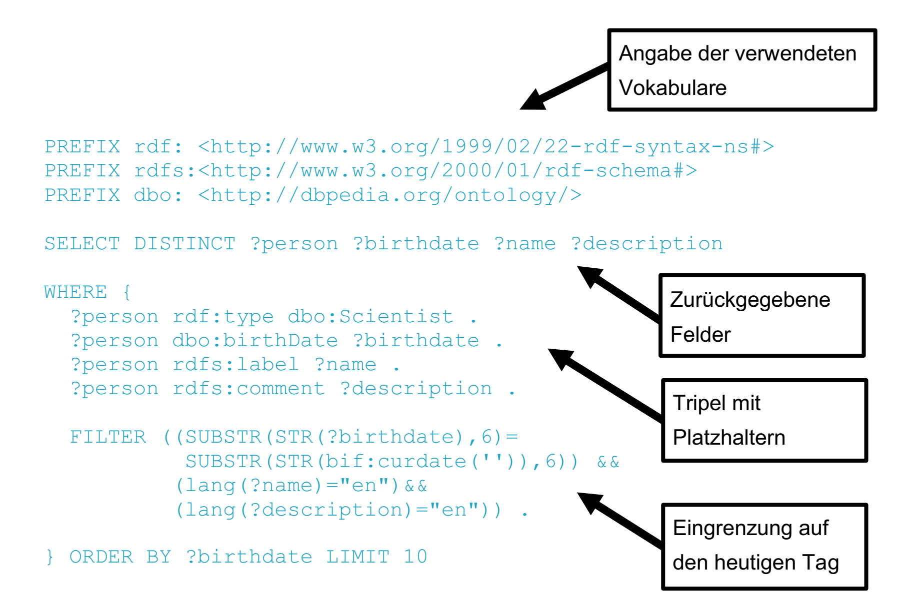


Abb. 3.7 SPARQL-Abfrage. Die Abfrage gibt bis zu zehn Wissenschaftler:innen zurück, die heute Geburtstag haben. Dazu werden drei Vokabulare (rdf, rdfs, dbo) verwendet. Sie können diese Abfrage unter OpenLink Software (2022; http://dbpedia.org/sparql) ausprobieren. Datengrundlage sind Wikipedia-Artikel. (Quelle: Eigene Darstellung auf Grundlage von Sack und Koutraki (2017))

Im Web frei verfügbare RDF-Daten werden als Linked Open Data (LOD) bezeichnet. Insbesondere in den Digital Humanities werden Datenbestände zunehmend als Linked Open Data verfügbar gemacht und mit anderen Datenbeständen verknüpft (Abb. 3.8). Im Kontext wissenschaftlicher Projekte eignet sich RDF erstens zur strukturierten Datenerfassung, etwa um systematisch bestimmte Wissensbereiche zu erfassen. Diese sogenannten Knowledge Graphs gruppieren sich um Entitäten wie Personen, Orte oder aber auch Softwares[32] und erfassen die Beziehungen zwischen den Entitäten in Form von Verwandtschaften, Einbindungen in Organisationen oder Erwähnungen in Texten. Zweitens können diese Daten abgefragt werden, um etwa Stichproben von Klöstern oder Schauspieler:innen zu ziehen (für ein Beispiel siehe Burggraaff und Trilling 2020).

Umfangreiche Datenbestände finden sich vor allem bei DBPedia und Wikidata. DBPedia extrahiert Daten aus Wikipedia-Artikeln. Umgekehrt werden struktu-

[32] Siehe zum Beispiel Schindler et al. (2021; https://data.gesis.org/somesci/) für die Erfassung von Softwares, die in wissenschaftlichen Publikationen erwähnt werden.

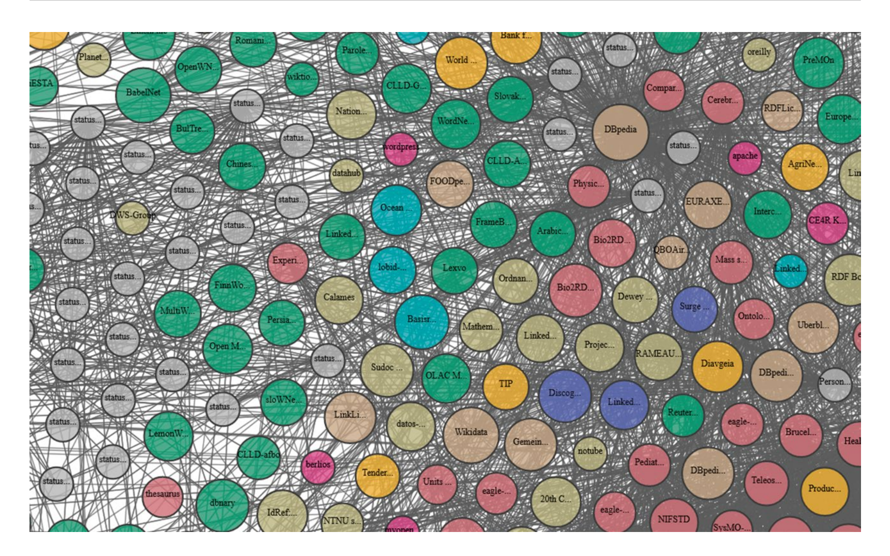


Abb. 3.8 The Linked Open Data Cloud. In der Linked Open Data Cloud wird erfasst, wie die Vokabulare und Datensätze verschiedener Anbieter zusammenhängen. (Quelle: McCrae (2021; https://lod-cloud.net))

rierte Daten aus Wikidata in die Wikipedia eingebunden. Geografische Datenbestände werden zum Beispiel durch GeoNames veröffentlicht.[33] RDF definiert dabei nicht, in welchem Format diese Daten abgespeichert werden – für diese sogenannte Serialisierung wird auf Formate wie JSON-LD, XML/RDF oder Textformate wie Turtle[34] aufgebaut. Viele Anbieter stellen mehrere Formate zur Verfügung (☛ *Repositorium*).

Im Web findet man RDF-kompatible Daten nicht nur als eigenständige Dateien, sondern auch direkt eingebettet in die Quelltexte von Webseiten (Abb. 3.9). RDF ist damit eine wichtige Grundlage für das Semantic Web,[35] womit das Ziel verfolgt wird, Daten miteinander zu verknüpfen und mit maschinenlesbarer Bedeutung zu versehen. Webseitenbetreiber können auf diese Weise strukturiert und standardi-

[33] Siehe Wick (2022; http://www.geonames.org/).

[34] Einige Formate gibt es in unterschiedlicher Komplexität: Turtle ist eine Untermenge von Notation3 und eine Übermenge von N-Triples.

[35] Eine Anlaufstelle zum Verständnis des Semantic Web ist W3C (2019; https://www.w3.org/2001/sw/wiki/). Zur Einbindung strukturierter Daten in Suchmaschinen siehe auch Google (2022b; https://developers.google.com/search/docs/advanced/structured-data/intro-structured-data).

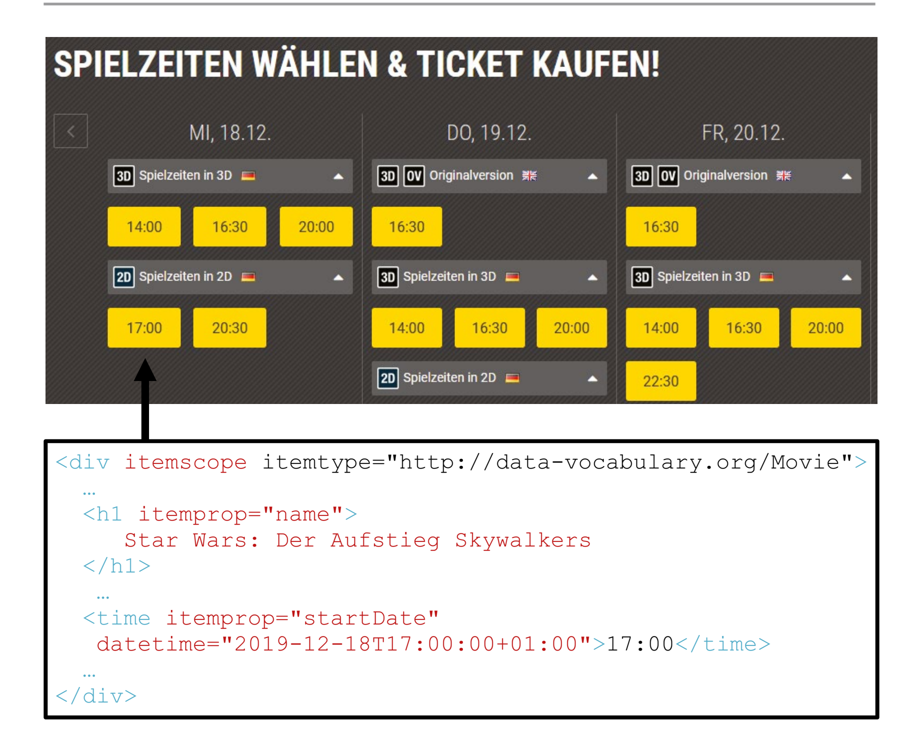


Abb. 3.9 Einbettung von strukturierten Daten in Webseiten mit Microdata. Das HTML-Element eines Films ist hier mit dem leeren Attribut `itemscope` markiert, alle untergeordneten Angaben beziehen sich darauf. Eigenschaften werden durch das `itemprop`-Attribut ausgezeichnet. Die Ausprägungen sind in den Elementen enthalten. (Quelle: CineStar (2019))

siert die Öffnungszeiten eines Unternehmens, Geburtsdaten historischer Personen, das Kinoprogramm, Angaben zu den Autor:innen einer Webseite und vieles mehr bereitstellen. Schaut man sich den Quelltext einer Webseite genauer an, begegnen einem insbesondere folgende Formate:

- In den Metadaten einer Webseite finden sich häufig Angaben mit standardisierten Vokabularen wie Dublin Core,[36] unter anderem für die Beschreibung der Autor:innen oder des Änderungsdatums einer Seite. Mit dem Open Graph Protocol[37] werden in den Metatags zudem Daten erfasst, die beispielsweise in Messengern beim Teilen von Links eine Vorschau der Webseite erzeugen.

---

[36] Siehe DCMI (2022; https://www.dublincore.org/).

[37] Siehe Facebook (2020; https://ogp.me/).

- Die bestehende HTML-Struktur kann um spezielle Attribute erweitert werden. RFDa[38] definiert Attribute wie `vocab`, `typeof` und `property` zur Einbettung von RDF-Daten. Eine Alternative ist Microdata,[39] das ebenfalls Attribute festlegt, die in den HTML-Code eingefügt werden. Benötigt werden vor allem `itemscope`, `itemprop` und `content`.
- Mikroformate wie hCard (Orte, Kontaktinformationen …) und hReview (Bewertungen von Büchern, Musik, Restaurants …) sind auf bestimmte Datensorten spezialisiert.[40] Sie verwenden die in HTML ohnehin definierten Standardattribute `class`, `rel` und `rev`. Mikroformate sind schnell zu erlernen und eignen sich damit für einfache Anwendungsfälle.

Einen Eindruck der Daten, die in eine Webseite eingebettet sind, geben Tools wie JSON-LD Playground.[41] Über Browser-Plugins[42] lassen sich die in eine Webseite eingebundene Daten auch direkt während des Surfens anzeigen und erkunden. Diese Daten können in der Regel in verschiedenen Formaten (= Serialisierungen von RDF) heruntergeladen werden. Ohne den Einsatz spezieller Tools können in Webseiten eingebettete Daten für wissenschaftliche Analysen über Webscraping (siehe Abschn. 7.1) ausgelesen werden.

Die verschiedenen Technologien und Datenbestände des Semantic Web erscheinen zunächst recht heterogen und unübersichtlich. Gute Anlaufpunkte für solche unübersichtlichen Themen sind sogenannte Awesome Lists – eine Zusammenstellung relevanter Themenaspekte bietet die Liste „Awesome Semantic Web“.[43] Dort finden Sie Verweise auf entsprechende wissenschaftliche Zeitschriften und erhalten Anregungen, inwiefern sich mit dieser Technologie wissenschaftliche Analysen durchführen lassen – das Resource Description Framework eröffnet eine Unmenge

[38] Eine Einführung in RDFa findet sich unter W3C (2015; https://www.w3.org/TR/xhtml-rdfa-primer/). Weitere Informationen siehe auch RDFa (2022; http://rdfa.info/).

[39] Microdata ist im HTML-Standard definiert, siehe Hickson et al. (2022; https://html.spec.whatwg.org/#microdata).

[40] Die Standardisierung findet über ein Wiki statt, siehe Microformats (2022; http://microformats.org/).

[41] Siehe JSON-LD (2022; https://json-ld.org/playground/). Ein weiteres Tool ist der HTML Structured Data Extractor (Herman 2012; https://www.w3.org/2012/sde/).

[42] Zum Beispiel der OpenLink Structured Data Sniffer (OpenLink Software 2021; https://osds.openlinksw.com/). Öffnen Sie beispielsweise die Seite der BBC mit klassischen Comedy-Programmen (https://www.bbc.co.uk/programmes/) und inspizieren Sie die Datensammlung über das Plugin.

[43] Siehe Semantalytics (2022; https://github.com/semantalytics/awesome-semantic-web).

an neuen Fragestellungen und Ansätzen zur Erfassung, Verknüpfung und Analyse von Wissen.

Übungsfragen

1. Was unterscheidet 1983 von „1983“ und warum ist dieser Unterschied wichtig?
2. Was legt die Zeichenkodierung einer Datei fest?
3. Mit welchen Mitteln können Markdown-Dateien formatiert werden?
4. Welche Trennzeichen werden in CSV-Dateien eingesetzt?
5. Wie finden Sie heraus, welches Trennzeichen in einer CSV-Datei verwendet wird? Überprüfen Sie die Beispiele im ☛ *Repositorium*!
6. Worin unterscheiden sich die Formate CSV, JSON und XML?
7. Was sind Unterschiede zwischen relationalen und nichtrelationalen Datenbanken?
8. Suchen Sie sich eine Datenbank (siehe Abschn. 2.3) und bestimmen Sie, ob es sich hierbei um eine relationale oder eine nichtrelationale Datenbank handelt!
9. Wie ist ein RDF-Tripel aufgebaut (☛ *Repositorium*)?
10. Was ist das Semantic Web?

Weiterführende Literatur

Russell, M. A. (2014). *Mining the social web. Data mining Facebook, Twitter, Linkedin, Google+, GitHub, and more.* (2. Aufl.). Sebastopol: O'Reilly.

Segaran, T., Evans, C. & Taylor, J. (2009). *Programming the Semantic Web.* Sebastopol: O'Reilly.

Silberschatz, A., Korth, H. F. & Sudarshan, S. (2020). *Database system concepts.* New York: McGraw-Hill.

Literatur

Ackoff, R. (1989). From data to wisdom. Presidential address to ISGSR, June 1988. *Journal of Applied Systems Analysis*, 16, 3–9.

Ballsun-Stanton, B. (2010). Asking about data: Experimental philosophy of Information Technology. In *5th International Conference on Computer Sciences and Convergence Information Technology, ICCIT 2010* (S. 119–124). Piscataway: Institute of Electrical and Electronics Engineers (IEEE). https://doi.org/10.1109/ICCIT.2010.5711041

Bray, T. (Internet Engineering Task Force IETF.). (2014). *The JavaScript Object Notation (JSON) Data Interchange Format*. Zugriff am 24.05.2022. https://datatracker.ietf.org/doc/html/rfc7159

Brickley, D. & Miller, L. (2014). *FOAF Vocabulary Specification 0.99*. Zugriff am 24.05.2022. http://xmlns.com/foaf/spec/

Burggraaff, C. & Trilling, D. (2020). Through a different gate: An automated content analysis of how online news and print news differ. *Journalism*, *21*(1), 112–129. https://doi.org/10.1177/1464884917716699

CineStar. (2019). *Star Wars: Der Aufstieg Skywalkers*. Zugriff am 01.06.2022. https://www.cinestar.de/kino-greifswald/film/star-wars-episode-ix

Cone, M. (2022). *Markdown Cheat Sheet. A quick reference to the Markdown syntax*. Zugriff am 24.05.2022. https://www.markdownguide.org/cheat-sheet/

DBpedia Association. (2021). *DBpedia*. Zugriff am 24.05.2022. https://www.dbpedia.org/

DCMI: Dublin Core Metadata Innovation. (2022). *Dublin Core*. Zugriff am 30.05.2022. https://www.dublincore.org/

Elastic. (2022). Elasticsearch (Version 8.2) [Computer software]. https://www.elastic.co/elasticsearch

Facebook. (2020). *The Open Graph protocol*. Zugriff am 02.07.2020. https://ogp.me/

Google. (2022a). Cloud Bigtable [Computer software]. https://cloud.google.com/bigtable

Google. (2022b). *Google Search Central. Understand how structured data works*. Zugriff am 30.05.2022. https://developers.google.com/search/docs/advanced/structured-data/intro-structured-data

Google, Microsoft, Yahoo & Yandex. (2022). *Schema.org*. Zugriff am 24.05.2022. https://schema.org/

Herman, I. (W3C Semantic Web, Hrsg.). (2012). *HTML Structured Data Extractor to RDF*. Zugriff am 30.05.2022. https://www.w3.org/2012/sde/

Hickson, I., Pieters, S., van Kesteren, A., Jägenstedt, P. & Denicola, D. (WHATWG, Hrsg.). (2022). *HTML. Living Standard*. Zugriff am 30.05.2022. https://html.spec.whatwg.org/multipage/

Holistics.io. (2022). *dbdiagram.io. Draw Entity-Relationship Diagrams, Painlessly*. Zugriff am 01.06.2022. https://dbdiagram.io/

ISO/IEC 2382. (2015). Information technology – Vocabulary.

JSON-LD. (2022). *JSON-LD Playground*. Zugriff am 30.05.2022. https://json-ld.org/playground/

Kluge, F. (2002). *Etymologisches Wörterbuch der deutschen Sprache* (24., durchges. und erw. Aufl.). Berlin: de Gruyter.

Link, V., Lohmann, S., Marbach, E., Negru, S. & Wiens, V. (2019). WebVOWL: Web-based Visualization of Ontologies (Version 1.1.7) [Computer software]. http://vowl.visualdataweb.org/webvowl.html

MariaDB Foundation. (2022). MariaDB Server. The open source relational database (Version 10.9.0) [Computer software]. https://mariadb.org/

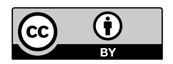
McCrae, J. P. (Insight Centre for Data Analytics, Hrsg.). (2021). *The Linked Open Data Cloud*. Zugriff am 01.06.2022. https://lod-cloud.net/

Meier, W. (2003). eXist: An open source native XML database. In G. Goos, J. Hartmanis, J. van Leeuwen, A. B. Chaudhri, M. Jeckle, E. Rahm et al. (Hrsg.), *Web, Web-Services, and Database Systems* (S. 169–183). Berlin: Springer. https://doi.org/10.1007/3-540-36560-5_13

Microformats. (2022). *Welcome to the microformats wiki!* Zugriff am 30.05.2022. https://microformats.org/wiki/Main_Page

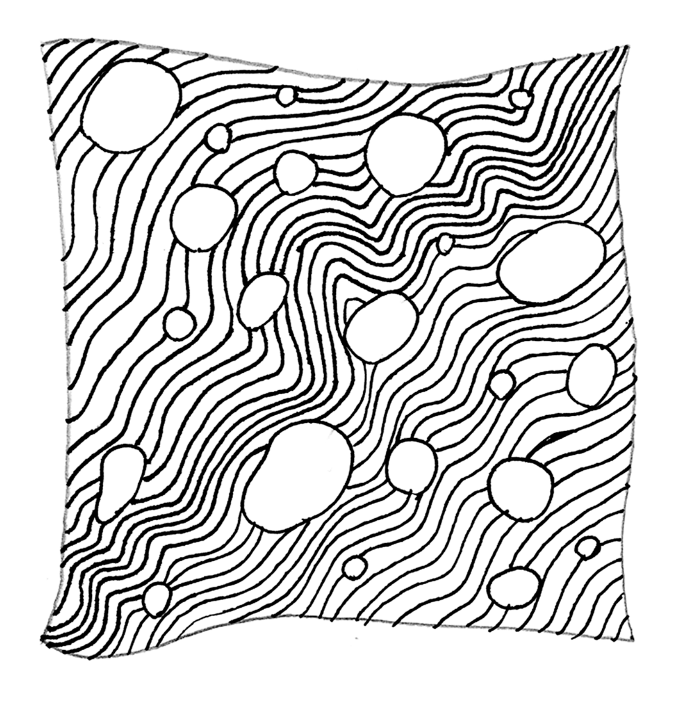
MongoDB. (2022) (Version 4.0.8) [Computer software]. https://www.mongodb.com/

Neo4j. (2022). Neo4j (Version 4.4.7) [Computer software]. https://neo4j.com/
OMG. (2022). *Unified Modeling Language (UML)*. Zugriff am 24.05.2022. https://www.uml.org
OpenLink Software. (2021). *The OpenLink Structured Data Sniffer (OSDS)*. Zugriff am 30.05.2022. https://osds.openlinksw.com/
OpenLink Software. (2022). *SPARQL Query Editor*. Zugriff am 01.06.2022. https://dbpedia.org/sparql/
Oracle. (2021). MySQL (Version 8.0) [Computer software]. https://www.mysql.com/
RDFa. (2022). *Linked Data in HTML*. Zugriff am 30.05.2022. http://rdfa.info/
Redis. (2022). Redis (Version 7.0) [Computer software]. https://redis.io/
Rowley, J. (2007). The wisdom hierarchy: representations of the DIKW hierarchy. *Journal of Information Science*, *33*(2), 163–180. https://doi.org/10.1177/0165551506070706
Ruttenberg, A. (2020). *Basic Formal Ontology (BFO)*. http://basic-formal-ontology.org/
Sack, H. & Koutraki, M. (2017). *Information Service Engineering,* openHPI. https://open.hpi.de/courses/semanticweb2017/
Schindler, D., Bensmann, F., Dietze, S. & Krüger, F. (2021). *SoMeSci – A 5 Star Open Data Gold Standard Knowledge Graph of Software Mentions in Scientific Articles*. Zugriff am 30.05.2022. https://data.gesis.org/somesci/
Semantalytics. (2022). *Awesome Semantic Web. A curated list of various semantic web and linked data resources.* Zugriff am 30.05.2022. https://github.com/semantalytics/awesome-semantic-web
Solid IT. (2022). *DB-Engines. Informationen zu relationalen und NoSQL Datenbankmanagementsystemen*. Zugriff am 24.05.2022. https://db-engines.com/
Text Encoding Initiative. (2022). *Text Encoding Initiative*. Zugriff am 24.05.2022. https://tei-c.org/
Unicode. (2021). *Submitting Emoji Proposals*. Zugriff am 24.05.2022. https://unicode.org/emoji/proposals.html
W3C. (2013). *Extensible Markup Language (XML) 1.0 (Fifth Edition). W3C Recommendation*. Zugriff am 24.05.2022. https://www.w3.org/TR/xml/
W3C. (2014). *RDF 1.1 Primer. W3C Working Group Note*. Zugriff am 24.05.2022. https://www.w3.org/TR/rdf11-primer/
W3C. (2015). *RDFa 1.1 Primer – Third Edition. Rich Structured Data Markup for Web Documents. W3C Working Group Note*. Zugriff am 30.05.2022. https://www.w3.org/TR/xhtml-rdfa-primer/
W3C. (2019). *Semantic Web*. Zugriff am 30.05.2022. https://www.w3.org/2001/sw/wiki/Main_Page
W3C. (2021). *Search engines*. Zugriff am 24.05.2022. https://www.w3.org/wiki/Search_engines
W3Schools. (2022a). *HTML Element Reference*. Zugriff am 24.05.2022. https://www.w3schools.com/tags/default.asp
Wick, M. (Unxos GmbH, Hrsg.). (2022). *GeoNames*. Zugriff am 24.05.2022. https://www.geonames.org/


BY



4 Datenextraktion



J. Jünger, C. Gärtner, *Computational Methods für die Sozial- und Geisteswissenschaften*, https://doi.org/10.1007/978-3-658-37747-2_4

Zusammenfassung

Dieses Kapitel führt in Verfahren zur Datenselektion- und transformation ein. Sie lernen, wie Sie mit regulären Ausdrücken Suchmuster formulieren, um unstrukturierte Daten aufzubereiten, wie Sie Elemente aus Webseiten mit CSS-Selektoren und XPath extrahieren und wie Sie Datenbanken mit SQL abfragen. Zudem erfahren Sie, wie Matrizen in verschiedene Formen gebracht werden und warum sich das lohnt.

Im Online-Repositorium unter https://github.com/strohne/cm finden Sie begleitend zum Kapitel weitere Materialien, auf die wir im Text mit ☛ verweisen.

Schlüsselwörter

Reguläre Ausdrücke · CSS-Selektoren · XPath · SQL · Wide- & Long-Format · Joins · Map-Reduce · Split-Apply-Combine · Matrizen · Matrixmultiplikation

Daten liegen in vielen unterschiedlichen Geschmacksrichtungen vor (siehe Kap. 3). Um daraus ein Gericht zuzubereiten – sie in eine Analyse einzuspeisen und konkrete Fragestellungen zu beantworten – müssen sie meist erst, wie in Abb. 4.1 veranschaulicht, zusammengestellt und aufbereitet werden. Dazu werden die für eine Analyse nötigen Daten zunächst aus dem gesamten Datenbestand selektiert (siehe Abschn. 4.1). Anschließend müssen die Daten noch bereinigt und fehlende Daten ergänzt werden. Schließlich werden sie so umgewandelt und strukturiert, dass sie ausgewertet werden können (siehe Abschn. 4.2). Das Ziel besteht meist darin, die verschiedenen Datenformate so zu manipulieren, dass Texte, Listen und Objekte am Ende rechteckig werden, sie also in tabellarische Form zu überführen. Für all diese Schritte, das Selektieren, Aufbereiten und Transformieren von Daten, werden nachfolgend einige Methoden und Werkzeuge vorgestellt.

Auch wenn dieses Schema eine erste Orientierung für die Datenextraktion geben kann, gestaltet sich der tatsächliche Ablauf einer Studie häufig anders. Im Verlauf einer Analyse lernt man immer weiter dazu und es kann sich herausstellen, dass man noch einmal einen Schritt zurückgehen muss, um die Daten anders auszuwählen. Auch sind diese Schritte mitunter ineinander verschachtelt. Wenn zum Beispiel während der Datenextraktion bereits eine Spracherkennung durchgeführt wird, um nur deutsch- oder englischsprachige Inhalte zu behalten, oder wenn nur Texte zu festgelegten Hashtags oder zu automatisiert erkannten Themen ausgewählt werden sollen, dann erfordert dies eine frühzeitige Aufbereitung und Analyse von Texten. Und auch während der Analyse beschäftigt man sich mitunter mit Ausschnitten der Daten, die noch einmal neu aufbereitet werden. Wichtig ist vor allem, dass man am Ende angeben kann, wie Erhebung, Auswahl, Aufbereitung

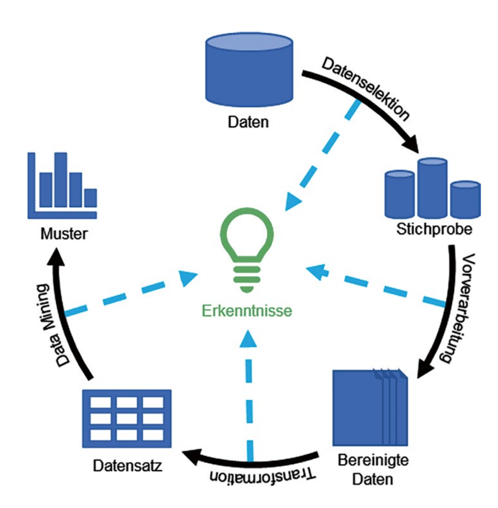


Abb. 4.1 Der Dataminingprozess. (Quelle: eigene Darstellung, in Anlehnung an Fayyad et al. 1996, S. 41)

und Analyse der Daten effektiv stattgefunden haben. Nur so lässt sich einschätzen, worauf sich die gewonnenen Erkenntnisse beziehen und inwiefern sie verallgemeinerbar sind.

4.1 Selektionsverfahren und -sprachen

Insbesondere schwach strukturierte Daten zeichnen sich dadurch aus, dass sie unübersichtlich sind und eine Unmenge irrelevanter Daten enthalten. Für die spätere Analyse werden deshalb zunächst die relevanten Datenpunkte aus dem Material extrahiert. Je nach Ausgangsformat eignen sich verschiedene Sprachen, um gezielt einzelne Daten anzusprechen und so zwischen Ausgangs- und Zielformat zu übersetzen. Im Folgenden werden vier solcher Sprachen vorgestellt.

1. Besonders flexibel einsetzbar sind **reguläre Ausdrücke**. Dabei handelt es sich um Ausdrücke mit Platzhaltern, die zum Durchsuchen beliebiger Texte verwendet werden können – dazu gehören auch die in Kap. 3 vorgestellten textbasierten Formate CSV, HTML, XML, JSON und SQL-Dumps. In Texten lassen sich mit regulären Ausdrücken etwa leicht Jahreszahlen identifizieren.
2. In den Markup-Formaten HTML und XML sind Texte hierarchisch bzw. baumartig vorstrukturiert. Um einzelne Elemente innerhalb dieser Strukturen zu finden, eignet sich insbesondere **XPath**. Die Sprache ist speziell für den Zugriff

auf XML-basierte Formate konzipiert worden und Teil der Datenbankabfragesprache XQuery.

3. **CSS-Selektoren** sind eine leichtgewichtige Alternative zu XPath, auch wenn Cascading Style Sheets (CSS) eigentlich für die optische Formatierung von Webseiten entwickelt wurden, etwa um die Schriftfarbe von Überschriften und die Abstände zwischen Absätzen festzulegen.
4. Die letzte in diesem Kapitel besprochene Sprache ist **SQL**, mit der Datenbanken abgefragt werden können. Hier geht es also weniger darum, unstrukturierte Daten in Form zu bringen, als vielmehr in bereits stark strukturierten Daten die relevanten Felder auszulesen.

4.1.1 Reguläre Ausdrücke

Computer arbeiten mit Zahlen bzw. mit digitalen Binärwerten. Dennoch lassen sich alle Datenformate als Text darstellen, denn auch jede Zahl lässt sich als Zeichen interpretieren. Welche Zahl für welches Zeichen steht, wird über eine sogenannte Zeichenkodierung (engl. *encoding*) festgelegt (siehe Kap. 3). So stehen die Zahl 67 in der ASCII-Kodierung für den Großbuchstaben „C“ und die Zahl 77 für den Buchstaben „M“. Erstaunlich viele Dateien sind intern als Textdateien aufgebaut und auch viele Programmiersprachen können Zahlen als Text interpretieren (Datentyp Zeichenkette).

Liegen Daten in Textdateien vor, können sie mit Editoren wir Atom oder Notepad++ geöffnet werden (siehe Kap. 1). Eine besondere Stärke von Texteditoren ist die Suchfunktion. Im einfachsten Fall gibt man dafür eine Zeichenkette ein und es werden nach und nach alle Textstellen angesprungen, die mit dem Suchausdruck übereinstimmen. Mit Texteditoren können in der Regel auch alle Fundstellen auf einmal markiert oder durch einen anderen Ausdruck ersetzt werden.

Besonders mächtig werden Suchfunktionen dann, wenn sie Platzhalter unterstützen, verbreitet sind das Fragezeichen für einzelne Zeichen und der Stern oder ein Prozentzeichen für beliebige aufeinanderfolgende Zeichen. Noch mehr Möglichkeiten bieten reguläre Ausdrücke. Reguläre Ausdrücke sind eine Sprache, um Suchmuster zu formulieren.[1] Sie kommen vor allem dann zur An-

[1] Reguläre Ausdrücke (engl. *regular expression* oder auch kurz *regex*) sind eine formale Sprache, eine Herleitung und Definition dieser Idee findet sich bei Kleene (1956). Im Gegensatz zur kreativen, natürlichen Sprache, die wir sprechen, folgen sie einer genau festgelegten und sehr eingeschränkten Grammatik. Reguläre Ausdrücke lassen sich in der sogenannten Chomsky-Hierarchie (Chomsky 1956) als Typ 3 verorten, darunter sind Sprachen mit einer regulären Grammatik zu verstehen. Das bedeutet höhere Typen und erst recht natürliche Sprachen können nicht in allen Facetten durch reguläre Ausdrücke beschrieben werden, aber sie eignen sich sehr gut für einfache sprachliche Muster, die durch finite Automaten erfasst werden können (Thompson 1968).

wendung, wenn unstrukturierte Daten aufbereitet werden müssen. Zum Beispiel lassen sich damit Links, Datumsangaben oder andere Daten aus dem Quelltext von Webseiten extrahieren. Aber auch bei der Datenmanipulation über Programmiersprachen wie R und Python lassen sich reguläre Ausdrücke einsetzen – beispielsweise, wenn in einer Tabelle neue Spalten mit URLs erstellt oder wenn beim Text Mining verschiedene Schreibweisen von Wörtern wie ausgeschriebene und nichtausgeschriebene Umlaute (ae vs. ä) berücksichtigt werden sollen.

Als Übungsmaterial für den Umgang mit regulären Ausdrücken eignet sich HTML-Quellcode von Webseiten (siehe Abschn. 3.4). Einen Einblick in den Quelltext einer Webseite erhält man in den meisten Browsern wie zum Beispiel Firefox, indem man mit der rechten Maustaste auf ein Element der Seite klickt und ELEMENT UNTERSUCHEN auswählt oder die Seite auf dem eigenen Computer abspeichert. Solche Quelltexte sehen wie folgt aus:[2]

```
<li class="ep-hover" data-jahr="201" data-kategorie="9"
data-land="158">
<a href="/real-humans-echte-menschen" title="Real
Humans - Echte Menschen (S 2012-)">
<span class="bold">Real Humans - Echte Menschen</span>
(<abbr title="Schweden">S</abbr> 2012-)</a>
</li>
<li class="ep-hover" data-jahr="201" data-kategorie="3"
data-land="184">
<a href="/real-husbands-of-hollywood" title="Real Husbands
of Hollywood (USA 2013-)"><span class="bold">Real
Husbands of Hollywood</span> (<abbr title="USA">USA</
abbr> 2013-)</a>
</li>
```

Im Beispiel sind Namen, Länder und Erscheinungsjahre von zwei Serien enthalten, die sich gut über reguläre Ausdrücke extrahieren lassen. Im Browser würde der Text wie folgt aussehen, enthält dann aber weniger Informationen:

```
Real Humans - Echte Menschen (S 2012-)
Real Husbands of Hollywood (USA 2013-)
```

[2] Quelle: Imfernsehen (2022; https://www.fernsehserien.de/serien-a-z/r).

Es fehlt beispielsweise die URL der Seite mit Detailinformationen zur Serie. Auch der im Browser angezeigte Text ist aus Maschinensicht unstrukturiert. Er müsste für die Datenverarbeitung zum Beispiel mit regulären Ausdrücken in eine Tabellenform überführt werden, bei der Name, Land und Datumsangaben durch ein einheitliches Trennzeichen getrennt sind (CSV-Dateien, siehe Abschn. 3.3). Kopieren Sie den Beispieltext in einen Texteditor und probieren Sie die folgenden Muster und eigene Variationen daran aus (☛ *Repositorium*)!

Platzhalter und Zeichenklassen Die einfachste Form eines regulären Ausdrucks besteht aus einer Zeichenfolge, die buchstäblich gesucht wird, zum Beispiel ein Wort:

```
Hollywood
```

Als Platzhalter können nun Zeichenklassen oder Zeichenbereiche verwendet werden. Der Punkt kennzeichnet ein beliebiges Zeichen, sodass folgender Ausdruck unter anderem die Zeichenfolgen „Bollywood“ oder „Mollywood“ findet:

```
.ollywood
```

Statt eines beliebigen Zeichens kann in eckigen Klammern ein Bereich angegeben werden:

```
[A-Z]ollywood
```

In diesen eckigen Klammern kann auch eine Liste möglicher Zeichen angegeben werden, zum Beispiel um auf „Bollywood“ und „Hollywood“ einzugrenzen:

```
[HB]ollywood
```

Mehrere Listen können kombiniert werden, etwa um Groß- und Kleinschreibung zu berücksichtigen. Denn reguläre Ausdrücke unterscheiden in der Standardeinstellung zwischen Groß- und Kleinschreibung:

```
[a-zA-Z]ollywood
```

Während bislang formuliert wurde, welche Zeichen enthalten sein dürfen, kann auch eine Negativliste definiert werden. Wenn direkt nach der öffnenden eckigen Klammer ein Caret ^ angegeben wird, dann werden alle Zeichen gefunden, außer den angegebenen. Das Beispiel findet also ausgerechnet „Hollywood“ nicht:

```
[^H]ollywood
```

Diese Technik lässt sich besonders gut in Kombination mit den im Folgenden besprochenen Quantoren einsetzen, um etwa alle Zeichen innerhalb doppelter Anführungszeichen zu identifizieren. Der folgende etwas kryptische Ausdruck findet Zeichenketten, die mit Anführungszeichen beginnen, dann mindestens ein anderes Zeichen beinhalten und schließlich mit einem Anführungszeichen enden:

```
"[^"]*"
```

Besonders häufig verwendete Zeichenklassen haben eigene Namen, sodass der Tippaufwand verringert wird (Tab. 4.1). Zum Beispiel lassen sich alle alphanumerischen Zeichen mit `\w` finden und Weißraum mit `\s`.

Für einige Fälle hilfreich sind zudem Modifikatoren, die das Gesamtverhalten des regulären Ausdrucks steuern. Damit kann unter anderem eingestellt werden, ob ein Suchausdruck Groß- und Kleinschreibung berücksichtigen oder auf mehrere Zeilen angewendet werden soll – die konkrete Umsetzung hängt vom verwendeten Texteditor bzw. dem Dialekt ab (siehe unten).

Tab. 4.1 Häufig verwendete Zeichenklassen

Klasse	steht für	umfasst
\d	[0-9]	Ziffern
\D	[^0-9]	alles außer Ziffern
\w	[a-zA-Z0-9_]	alphanumerische Zeichen und Unterstrich
\W	[^a-zA-Z0-9_]	alles außer alphanumerischen Zeichen und Unterstrich
\s	[\r\t\n\f]	Weißraum (engl. *whitespace*)
\S	[^ \r\t\n\f]	Kein Weißraum (engl. *whitespace*)

Quelle: Eigene Darstellung

Quantoren Bislang wurden immer nur Platzhalter für einzelne Zeichen formuliert. In der Kombination mit Quantoren lassen sich auch Platzhalter für längere Zeichenfolgen festlegen. Ein Quantor gibt an, wie häufig ein Zeichen mindestens und wie häufig es höchstens vorkommen darf. Der folgende Suchausdruck findet beispielsweise alle vierstelligen Zahlen:

```
[0-9]{4,4}
```

Zunächst wird hier in eckigen Klammern festgelegt, welche Zeichen vorkommen dürfen. In den geschweiften Klammern wird dann vor dem Komma die Mindestanzahl und nach dem Komma die Höchstanzahl der erlaubten Vorkommen definiert. Hier werden also Zeichenausdrücke gefunden, die mindestens vier und gleichzeitig höchstens vier aufeinanderfolgende Ziffern enthalten. Wenn wie hier Minimum und Maximum identisch sind, kann der Ausdruck vereinfacht werden. Es reicht die Angabe der gewünschten Anzahl:

```
[0-9]{4}
```

Man kann die Maximalbedingung auch weglassen, etwa um alle Zahlen mit mindestens zwei Ziffern zu finden, die aber beliebig lang sein dürfen:

```
[0-9]{2,}
```

Für drei besonders häufig genutzte Quantoren gibt es spezielle Symbole. Das Fragezeichen entspricht der Bedingung `{0,1}` und findet alle Zeichenketten, in denen ein Zeichen keinmal oder höchstens einmal vorkommt. Das folgende Beispiel findet deshalb sowohl den Singular „Jahr“ als auch den Plural „Jahre“:

```
Jahre?
```

Das Pluszeichen wird verwendet, wenn ein Zeichen mindestens einmal vorkommen soll, zum Beispiel für beliebig lange Zahlen:

```
[0-9]+
```

Das Sternchen ist am lockersten und lässt beliebige Häufigkeiten zu, es ist keine Minimal- oder Maximalanzahl festgelegt. Das ist in Kombination mit Zeichenklassen (z. B. in eckigen Klammern) sehr mächtig. Der folgende Ausdruck findet alle Wörter, die mit „Jahr“ beginnen. Dazu zählen das Wort „Jahr“ selbst und Wör-

ter wie „Jahreszahl“, „Jahre“ oder „Jahres“. Die Suche endet erst dann, wenn kein kleiner Buchstabe mehr folgt (sondern zum Beispiel ein Leerzeichen):

```
Jahr[a-z]*
```

Quantoren sind normalerweise gierig (engl. *greedy*), das heißt die Suche wird so lange fortgesetzt, wie es geht. Soll die Suche dagegen so früh wie möglich abgebrochen werden, kann das Suchverhalten umgeschaltet werden, indem an einen Quantor ein Fragezeichen angehängt wird.

Maskierung reservierter Zeichen und Sonderzeichen Viele Zeichen haben innerhalb regulärer Ausdrücke also eine bestimmte Bedeutung, zum Beispiel Fragezeichen, Punkt, Schrägstrich oder auch Klammern. Was macht man nun, wenn man genau diese Zeichen finden will? Die Zeichen werden in diesen Fällen maskiert (engl. *escaped*), indem ein Backslash vorangestellt wird. Damit steht der Punkt im folgenden Beispiel tatsächlich für einen Punkt und nicht für ein beliebiges Zeichen.

```
3\. Januar
```

Nun verschiebt sich das Problem auf den Schrägstrich. Die Lösung bleibt gleich: Der Schrägstrich wird durch einen Schrägstrich maskiert. Sollen zwei Schrägstriche hintereinander gefunden werden, müssen beide maskiert werden, die Anzahl der Schrägstriche wird also verdoppelt.

Eine Besonderheit ergibt sich, wenn mit regulären Ausdrücken in R oder Python gearbeitet werden soll (siehe Kap. 5). Da der Schrägstrich innerhalb dieser Sprachen bereits eine Bedeutung hat, muss er maskiert werden, damit er überhaupt als Maskierungszeichen erkannt wird.[3] In R ist also der folgende Ausdruck identisch mit dem vorangegangenen Beispiel, wobei reguläre Ausdrücke in vielen Programmiersprachen als Zeichenkette gelten und zusätzlich in Anführungszeichen gesetzt werden:

```
"3\\. Januar"
```

[3] Der Grund liegt darin, dass zunächst der Quelltext des Skripts vom Compiler oder Interpreter der Programmiersprache ausgewertet wird. Erst danach wird der Ausdruck an die Funktionen zum Verarbeiten (d. h. Parsen) eines regulären Ausdrucks übergeben.

Tab. 4.2 Sonderzeichen

Zeichen	Bedeutung
`\n`	Zeilenumbruch (Line Feed)
`\r`	Zeilenumbruch (Carriage Return)
`\t`	Tabulator

Quelle: Eigene Darstellung

Die Maskierung der Maskierung mag etwas verwirrend sein. Unter Python gibt es die Möglichkeit, Zeichenketten mit einem Präfix als raw-String zu kennzeichnen.[4] Dann werden Backslashes nicht als Maskierungszeichen gewertet und müssen nicht doppelt maskiert werden:

```
r'3\. Januar'
```

Der Schrägstrich wird auch eingesetzt, um Sonderzeichen zu markieren. Dazu gehören nicht sichtbare Zeichen wie Zeilenumbrüche oder Tabulatoren (Tab. 4.2).

Unter Windows werden Zeilenumbrüche in Textdateien meistens durch eine Kombination aus Line Feed und Carriage Return (`\r\n`) gekennzeichnet, während in anderen Betriebssystemen ein einfaches Line Feed (`\n`) ausreicht. Die Suche nach Zeilenumbrüchen ist nur selten notwendig, zum Beispiel, wenn sie über Suchen & Ersetzen gelöscht werden sollen. Auf diese Weise können erst alle Zeilenumbrüche entfernt werden, um dann gezielt an bestimmten Stellen neue Zeilenumbrüche einzufügen. Wenn das Ersetzen nicht im Vordergrund steht, ist es aber einfacher, mit Ankern zu arbeiten (siehe unten).

Wenn Unicode unterstützt wird (siehe Kap. 3), dann können Sonderzeichen auch über eines der folgenden beiden Muster angegeben werden:

```
\u00A0
```

```
\x{00A0}
```

In diesen Mustern steht die Folge `00A0` für die Unicode-Nummer des geschützten Leerzeichens. Bei der Arbeit mit echten Texten stößt man immer wieder

[4] Siehe Kuchling (2017; https://docs.python.org/3.3/howto/regex.html#the-backslash-plague) und Python Software Foundation (2022a, https://docs.python.org/3/reference/lexical_analysis.html#string-and-bytes-literals).

auf Situationen, in denen Sonderzeichen nicht auf den ersten Blick erkennbar sind. So wie zwischen Groß- und Kleinschreibung unterschieden wird, lassen sich auch nicht alle Arten von Leerzeichen über ein einfaches Leerzeichen oder alle Arten von Bindestrichen über ein Minuszeichen finden. Im zu Beginn des Kapitels angegebenen Quelltext kommen sowohl geschützte Leerzeichen (engl. *non breaking space*) als auch ein Halbgeviertstrich (engl. *en dash*) vor. Im Zweifelsfall kann man die Zeichen direkt aus dem Text herauskopieren und in einen regulären Ausdruck übernehmen.

Der Kontext von Ausdrücken: Anker und Lookarounds Ohne weitere Eingrenzung finden reguläre Ausdrücke Muster an beliebigen Stellen im Text. Manchmal ist es aber hilfreich, nur den Anfang oder das Ende eines Texts oder eine Zeile zu durchsuchen. Das Muster wird dafür mit einem Caret `^` am Anfang oder mit einem Dollarzeichen `$` am Ende verankert. Beides kann kombiniert werden. In der Regel wird jede Zeile einzeln behandelt, sodass folgendes Suchmuster nur Zeilen finden würde, die nichts anderes als das Wort „Jahr“ enthalten:

```
^Jahr$
```

Eine weitere Möglichkeit, den Kontext eines gesuchten Texts zu berücksichtigen, sind Lookarounds. Hierbei wird angegeben, was in der Umgebung des gesuchten Texts vorkommen oder nicht vorkommen darf. Diese Muster gibt es einmal als vorausschauende (engl. *look ahead*) oder zurückschauende (engl. *look behind*) Muster und einmal als positive oder negative Muster. Ein positiver Lookahead würde nur Zeichenketten finden, auf die das definierte Muster folgt:

```
[0-9]+(?= Stunden)
```

Hiermit würden Zahlen gefunden werden, auf welche ein Leerzeichen und das Wort „Stunden“ folgen. Das folgende Muster wird also in runde Klammern gesetzt und zu Beginn mit `?=` markiert. Ein negativer Lookahead findet Text, auf den *nicht* das angegebene Muster folgt, zum Beispiel alle Zahlen, die keine Stundenangaben sind. Anstelle des Gleichheitszeichens wird ein negativer Lookahead mit einem Ausrufezeichen[5] markiert:

```
[0-9]+(?! Stunden)
```

[5] Ausrufezeichen werden in vielen Programmiersprachen zur Negation eingesetzt.

Nach dem gleichen Prinzip lassen sich Zahlen finden, die auf einen bestimmten Text folgen. Das Muster wird wiederum in Klammern gesetzt und mit einem Fragezeichen eingeleitet, bei Lookbehinds folgt darauf aber ein Kleinerzeichen. Auf diese Weise können etwa Größenangaben gefunden werden:

```
(?<=Größe )[0-9]+
```

Ein negativer Lookbehind wird wieder mit einem Ausrufezeichen anstelle des Gleichheitszeichens formuliert und findet dann alle Zahlen, vor denen das angegebenen Muster *nicht* steht:

```
(?<!Größe )[0-9]+
```

Im ersten Moment erscheinen Lookarounds nicht nur kompliziert, sondern vielleicht auch unsinnig. Wenn man zum Beispiel eine Stundenangabe finden will, dann kann man das auch ohne Lookahead bewerkstelligen, das heißt ohne die zusätzlichen Klammern, das Fragezeichen und das Gleichheitszeichen. Dann allerdings wird nicht nur die Zahl gefunden, sondern auch die Einheit bzw. das Wort. Gerade wenn man Daten extrahieren oder durch andere Daten ersetzen will (Suchen & Ersetzen), kann man sich mit Lookarounds auf die wesentlichen Daten konzentrieren. Sie sind am Anfang möglicherweise etwas schwer zu verstehen, in der einen oder anderen Situation aber doch sehr hilfreich.

Alternativen, Gruppen und Ersetzungen Lookarounds setzen voraus, dass ein Teil des Musters in runde Klammern gesetzt wird. Damit wird der entsprechende Teil des Musters gruppiert. Solche Gruppen können in Kombination mit einem senkrechten Strich `|` (auch Pipe genannt) zudem dazu genutzt werden, um Alternativen zu formulieren. Die Pipe kann also als „oder" übersetzt werden. Das folgende Muster findet alle Zeichenketten, die mit einer Zahl beginnen und dann getrennt durch ein Leerzeichen mit einem der Wörter „Stunde", „Minute" oder „Sekunde" fortgesetzt werden:

```
[0-9]+ (Stunde|Minute|Sekunde)
```

Gruppierungen können verschachtelt werden, das heißt, eingeklammerte Ausdrücke können bei Bedarf auch noch einmal eingeklammert werden, um so Alternativen innerhalb von Alternativen zu formulieren.

Aufeinanderfolgende Zeichen werden auch ohne Umklammerung als Gruppierung erkannt werden. Dies liegt an der Bindungskraft, die zwischen Zeichen besteht und die dafür verantwortlich ist, dass die Zeichen in einer gewissen Reihenfolge abgearbeitet werden. Dies folgt der gleichen Logik wie sie bei grundlegenden Rechenoperationen vorliegt. Beispielsweise gilt in der Mathematik die Regel Punkt- vor Strichrechnung: $6 \times 5 + 3 = 33$. Mit Klammern lässt sich die Reihenfolge und somit das Ergebnis verändern: $6 \times (5 + 3) = 48$. Bei regulären Ausdrücken werden standardweise aufeinanderfolgende Buchstaben oder Zahlen als Einheit gesehen. Muster in regulären Ausdrücken werden nach folgender Priorität berücksichtigt: Klammern vor Quantoren vor Sequenzen vor Ankern und schließlich vor Alternativen.

Gruppen sind vor allem dann hilfreich, wenn Teile eines Texts ersetzt werden sollen. Ein Texteditor wie Notepad++ erlaubt nicht nur die Formulierung von Suchmustern, sondern ebenso die Formulierung von Ersetzungen. In dem Ersetzungstext kann durch Einklammern auf einzelne Teile der Ausdrücke Bezug genommen werden. Dazu werden die runden Klammern von links nach rechts durchgezählt (immer nur die öffnende Klammer). So findet das folgende Beispiel eine Datumsangabe in dem in Europa verbreiten Format: TT.MM. JJJJ. Um nun einzelne Teile adressieren zu können, werden zunächst im Suchmuster Teile durch runde Klammern gruppiert, zusätzlich müssen die Punkte maskiert werden:

```
([0-9]{2})\.([0-9]{2})\.([0-9]{4})
```

Sollen nun einzelne Einheiten ersetzt oder vertauscht werden, dann wird dazu im Ersetzungsausdruck ein Backslash gefolgt von der Nummer der Klammer eingefügt. Wenn im Beispiel die Datumsangabe durch JJJJ.MM.TT. ersetzt werden soll, können die einzelnen Gruppen wie folgt vertauscht werden.

```
\3.\2.\1
```

Bei Ersetzungen müssen Punkte und andere Zeichen nicht maskiert werden. Der Vollständigkeit halber sei angemerkt, dass auf den gesamten gefundenen Text mit der Nummer `\0` Bezug genommen werden kann.

Dialekte Für das Erlernen von regulären Ausdrücken ist wie bei jedem Lernen von Sprachen viel Übung nötig. Die bislang angesprochenen Möglichkeiten stellen den

Kern von regulären Ausdrücken dar und sollten ausreichen, um sich am Anfang zurechtzufinden. Weitere Möglichkeiten finden Sie in vielzähligen Tutorials.[6] Ein Tool, das beim Zusammenstellen von regulären Ausdrücken behilflich sein kann, ist RegExr.[7] Hier können Sie zunächst einen unstrukturierten Text eingeben und anschließend reguläre Ausdrücke formulieren, die Ergebnisse werden im Text markiert.

Zu beachten ist dabei, dass verschiedene Programme – fast so wie bei menschlichen Sprachen – verschiedene Dialekte sprechen. Am häufigsten sind POSIX- oder PCRE-kompatible Dialekte anzutreffen. Die jeweils korrekte Syntax sollte man in der Dokumentation der jeweiligen Software nachschlagen. In Notepad++ und R kommt PCRE zum Einsatz, in Python wird ein PCRE-ähnlicher Dialekt verwendet. Bevor Sie nun aber die Dokumentationen wälzen, probieren Sie erst einmal selbst aus, wie weit Sie mit den hier angeführten Möglichkeiten kommen.

Anwendungsbeispiel: URLs extrahieren Reguläre Ausdrücke werden für das Extrahieren von Daten zwar häufig in Kombination mit Programmiersprachen wie R oder Python eingesetzt (siehe Kap. 5). Aber auch in Texteditoren lässt sich mit etwas Kreativität viel erreichen. Sie können damit zum Beispiel alle URLs aus einem Dokument auslesen. Im folgenden Beispiel geht es um die Frage, wie eine Webseite mit anderen Online-Diensten vernetzt ist. Grundsätzlich sind verschiedene Webseiten über URLs verlinkt, die im Quelltext zu finden sind (siehe Abschn. 2.1).

Eine Möglichkeit, die verlinkten Seiten zu identifizieren, bestünde also in der Analyse des Quelltexts ähnlich wie beim Webscraping. Gerade bei dynamischen Seiten werden aber auch noch später durch Skripte weitere Inhalte nachgeladen (siehe Abschn. 7.1). Um diese Abfragen nachgeladener Inhalte zu sehen, öffnen Sie im Browser die Entwicklerkonsole (in der Regel mit der Taste F12) und wechseln Sie in den Reiter Netzwerkanalyse (Abb. 4.2). Wenn Sie nun eine Webseite aufrufen, werden alle Zugriffe auf weitere Ressourcen im unteren Bereich protokolliert. Eine Webseite setzt sich meistens aus sehr vielen Einzelteilen zusammen; der Aufruf der Seite https://www.golem.de zieht zum Beispiel aktuell über 600 Abfragen nach sich. Wenn Werbeblocker deaktiviert sind, besteht ein Teil davon aus Werbeeinblendungen wie in der Abbildung am rechten Rand zu sehen ist. Mit der Funktion Netzwerkanalyse können Sie nachvollziehen, welche Tracking-Dienste oder externe Werbeinhalte in eine Seite eingebunden sind.

[6] Siehe zum Beispiel unter Goyvaerts (2021; https://regular-expressions.info).

[7] Siehe Skinner (2021; https://regexr.com/).

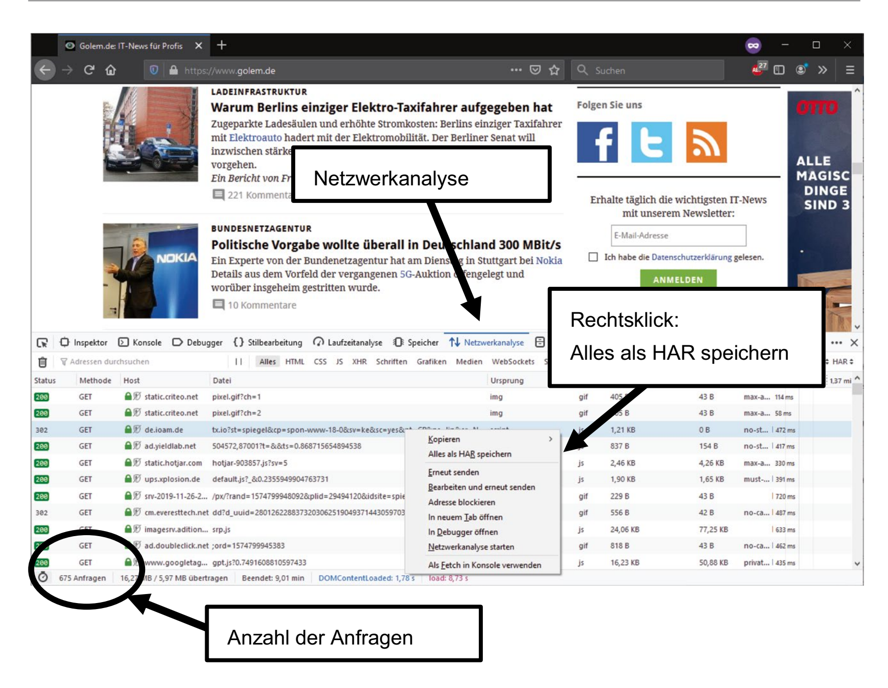


Abb. 4.2 Hintergrundabfragen einer Webseite. (Quelle: eigene Darstellung; Golem (2022; https://www.golem.de))

Um diese Daten weiterzuverarbeiten, können sie mit einem Rechtsklick als HAR-Format abgespeichert werden. Das HAR-Format ist ein JSON-Format (siehe Abschn. 3.5) und kann mit entsprechenden Programmen weiterverarbeitet werden, aber auch mit einem einfachen Texteditor. Öffnen Sie die exportierte Datei anschließend mit einem Texteditor (siehe Kap. 1). In den ersten Zeilen werden allgemeine Daten zur Abfrage angegeben. Hinter dem Wert `entries` folgt eine Liste aller Abfragen. Im Schlüssel `request.url` ist für jede Abfrage die URL angegeben.

Da jede dieser Zeilen gleich formatiert ist, können Sie einfach danach suchen (Abb. 4.3). Das folgende Suchmuster findet alle Zeilen, die mit einer beliebigen Anzahl von Leerzeichen beginnen und dann den Schlüssel `"url"` (inklusive der Anführungszeichen) enthalten, sofern Sie im Texteditor den Suchmodus auf reguläre Ausdrücke einstellen:

```
^ *"url"
```

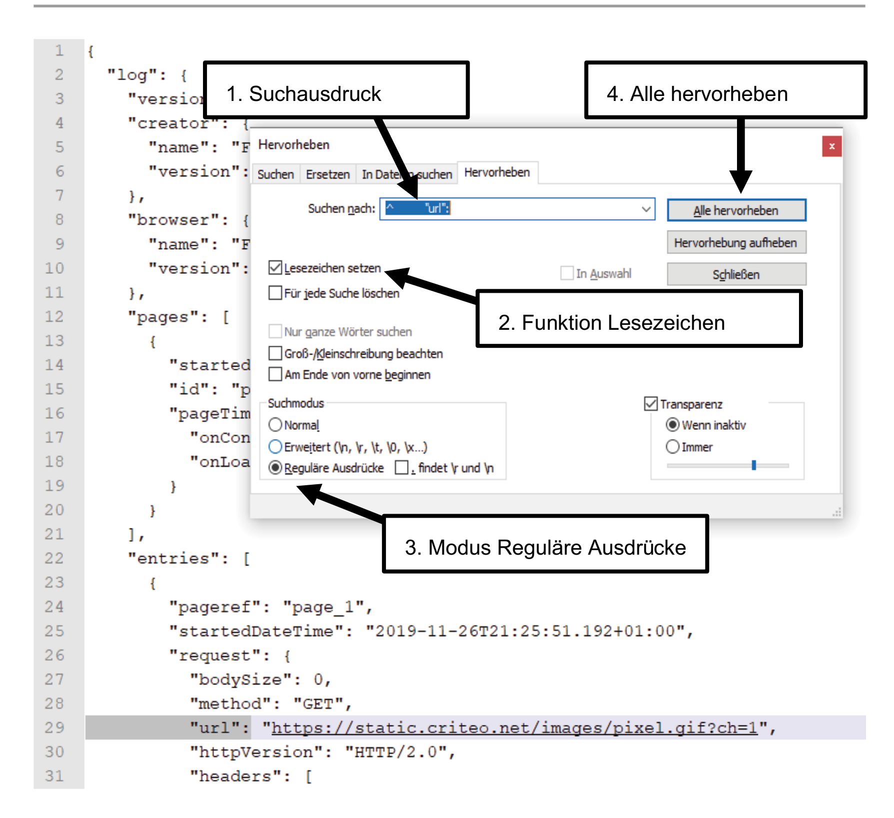


Abb. 4.3 Zeilen mit regulären Ausdrücken in Notepad++ hervorheben. (Quelle: eigene Darstellung)

Mit dem Caret ^ wird der Text am Zeilenanfang verankert (siehe oben), sodass nur die Zeilen gefunden werden, die mit dem Ausdruck beginnen.[8]

Wenn Sie nun eine Suche ausführen, sollten alle Zeilen mit den URLs angesprungen werden. Damit diese Zeilen weiterverarbeitet werden können, wechseln Sie zum Beispiel in Notepad++ in den Reiter HERVORHEBEN, aktivieren Sie die Funk-

[8] Diese Vorgehensweise funktioniert nur dann, wenn die gesuchten Texte in einzelnen Zeilen stehen und die Zeilen sich eindeutig identifizieren lassen. Sollte das nicht der Fall sein, müssen Sie mit Suchen & Ersetzen erst Zeilenumbrüche vor und nach der gesuchten Zeichenkette einfügen. Für die Suche nach URLs werden die nötigen regulären Ausdrücke allerdings sehr komplex, siehe zum Beispiel Silva (2008; https://stackoverflow.com/questions/161738/what-is-the-best-regular-expression-to-check-if-a-string-is-a-valid-url). In diesen Fällen ist eine Verarbeitung von JSON mit R oder Python günstiger.

tion LESEZEICHEN SETZEN und klicken Sie schließlich auf die Schaltfläche ALLE HERVORHEBEN. Im SUCHEN-Menü (in der oberen Menüleiste, außerhalb der Suchfunktion) finden Sie weit unten einen Punkt LESEZEICHEN mit verschiedenen Optionen. Sie können hier zum Beispiel alle Zeilen mit Lesezeichen in die Zwischenablage kopieren. Um damit weiterzuarbeiten, legen Sie mit Notepad++ ein neues Textdokument an und fügen Sie den Inhalt der Zwischenablage ein – so haben Sie mit einfachen Mitteln alle Drittanbieter identifiziert, auf die eine Webseite zurückgreift.

4.1.2 CSS-Selektoren

Reguläre Ausdrücke sind mächtig, wenn es um die Behandlung von unstrukturiertem Text geht. Allerdings ist die Formulierung von regulären Ausdrücken in einigen Fällen auch aufwendig und fehleranfällig. Wenn beispielsweise aus einem HTML-Quelltext ein bestimmtes Element extrahiert werden soll, das selbst wieder Tags enthält, ist es nahezu unmöglich, einen verständlichen und dennoch alle Spezialfälle umfassenden Ausdruck zu formulieren. Das folgende Beispiel wäre ein solcher Fall, wenn man das komplette `<li>`-Element mit allen Inhalten inklusive der Unterelemente erfassen will:

```
<li class="ep-hover" data-jahr="201" data-kategorie="9"
data-land="158">

        <ahref="/real-humans-echte-menschen" title="Real
        Humans - Echte Menschen (S 2012-)">

        <span class="bold">Real Humans - Echte Menschen
        </span> (<abbr title="Schweden">S</abbr> 
        2012-)</a>

</li>
```

Handelt es sich um Text, der einem definierten Format wie XML oder HTML folgt, besteht eine bessere Lösung darin, den Text mit einem darauf spezialisierten Parser einzulesen. Ein Parser erkennt einzelne Elemente und reproduziert die Struktur der Dokumente so, dass darauf gezielt zugegriffen werden kann. Der Quelltext ist danach in einem sogenannten Document Object Model (DOM) erfasst. Die direkt untergeordneten Elemente werden im DOM als Kindelemente

(engl. *children*) bezeichnet, alle untergeordneten Elemente, also auch Kinder von Kindern, als Nachfahren (engl. *descendants*). Die direkt übergeordneten Elemente heißen Eltern (engl. *parents*) und die Kette aller übergeordneten Elemente Vorfahren (engl. *ancestors*). So kann beispielsweise das -Element nicht nur als Abfolge der Zeichen <, l, i sowie > eingelesen, sondern vielmehr als Element mit den Kindelementen <a> sowie <span> geparsed werden. In vielen Programmiersprachen gibt es Parser, die eine Navigation in der DOM-Struktur ermöglichen, zum Beispiel verarbeitet das Python-Modul Beautiful Soup die Formate HTML und XML (Richardson 2022; siehe Abschn. 7.1).

Einzelne Elemente im DOM können anschließend über CSS-Selektoren identifiziert werden. CSS steht für Cascading Style Sheets und wird normalerweise dazu verwendet, die Darstellung von HTML-Seiten vorzugeben. In einer CSS-Datei kann etwa wie folgt festgelegt werden, dass alle Links grün erscheinen:

```
a { color:green; }
```

Der erste Teil dieser Angabe ist ein CSS-Selektor, der alle a-Elemente adressiert. Es folgt in geschweiften Klammern die Angabe der Formatierung. Für die Datenerhebung ist die Formatierung unwichtig, aber wenn ein Text etwa mit Beautiful Soup geparsed wurde, können über CSS-Selektoren einzelne Elemente zur Extraktion angesteuert werden.

Der einfachste CSS-Selektor benennt wie im Beispiel einfach den Tag-Namen – welcher im Quellcode direkt auf die öffnende spitze Klammer < folgt. Bezeichner, die mit einem Punkt beginnen, selektieren alle Elemente mit einem bestimmten Klassen-Attribut. Bezeichner mit einem Doppelkreuz # selektieren die Elemente über eine ID. Klassen- und ID-Attribute finden sich hinter dem Tag-Namen und sind nach dem Muster class="bezeichnung" bzw. id="bezeichnung" angegeben. Diese Notationsvarianten lassen sich auch kombinieren (Tab. 4.3). Wenn der Elementname, eine oder mehrere Klassen oder eine ID direkt ohne Leerraum aneinander gekettet werden, dann müssen alle diese Merkmale auf das ausgewählte Element zutreffen.

Um dagegen in die Hierarchie eines Dokuments einzutauchen, werden mehrere der Selektoren nacheinander notiert. Dadurch werden nur diejenigen Elemente selektiert, die entsprechende Vorfahren haben. Soll ein Element ein direktes Kindelement sein, wird dies mit dem >-Zeichen zwischen den Angaben ausgedrückt. Ohne >-Zeichen werden auch alle Nachfahren adressiert, die nicht unmittelbare Kinder des Elternelements sind. Um einzelne Elemente zu erfassen, hangelt man sich also im ersten Schritt an der Hierarchie entlang (Abb. 4.4). Wenn

Tab. 4.3 Beispiele für CSS-Selektoren

Selektor	Bedeutung	Beispiel
a	Alle a-Elemente	Real Humans
.bold	Alle Elemente mit der Klasse „bold“	Real Humans
.bold.hover	Alle Elemente mit den beiden Klassen „bold“ und „hover“	Real Humans
#footer	Das Element mit der ID „footer“	<div id="footer">5 Ergebnisse</div>
a.extern	Alle a-Elemente mit der Klasse „extern“	Example
p#teaser	Alle p-Elemente mit der ID „teaser“	<p id="teaser">Eine Einleitung leitet ein.</p>
p a	Alle a-Elemente innerhalb von p-Elementen	<p id="teaser">Eine **Einleitung** leitet ein.</p>
p > a	Alle a-Elemente, die direkt einem p-Element untergeordnet sind	<p id="teaser">Eine **Einleitung** leitet ein.</p>

Quelle: eigene Darstellung

die Elemente dann auch ohne Berücksichtigung der Hierarchiestufen eindeutig identifizierbar sind, kann der Ausdruck vereinfacht werden. Im Beispiel, solange es nur ein einziges span-Element gibt, würden die folgenden vier Ausdrücke alle das Gleiche leisten:

```
li > a > span.bold
li a span.bold
span.bold
span
```

Die Beispiele umfassen die wichtigsten Fälle von CSS-Selektoren. Es sind darüber hinaus deutlich komplexere Ausdrücke möglich, bei denen etwa die Anzahl vorangegangener Elemente, der Textinhalt eines Attributs oder weitere Bedingungen erfüllt sein müssen.[9]

[9] Für eine detailliertere Einführung in CSS-Selektoren, mit der Option zum Ausprobieren, siehe beispielsweise Selfhtml (2022; https://wiki.selfhtml.org/wiki/CSS/Tutorials/Selektoren).

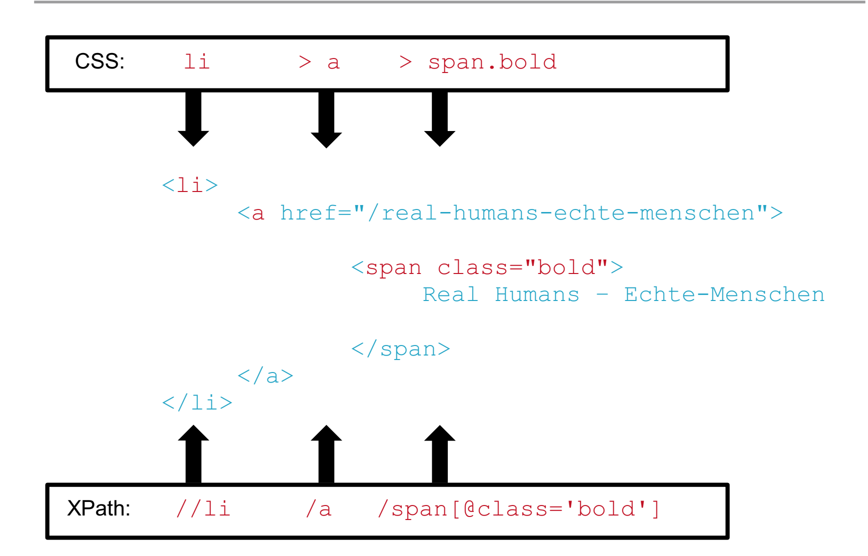


Abb. 4.4 Die hierarchische Struktur von CSS-Selektoren (oben) und XPath (unten) im Vergleich. (Quelle: eigene Darstellung)

CSS-Selektoren spielen beim Webscraping eine wichtige Rolle (siehe Kap. 7) und funktionierende Ausdrücke können über die Entwicklerkonsole des Browsers (aufrufbar über die Taste F12) gefunden werden. Steuern Sie dafür das gewünschte Element an und klicken Sie mit der rechten Maustaste auf die entsprechende Zeile in der Entwicklerkonsole, dort kann ein Selektor in der Regel über Copy Selector in die Zwischenablage kopiert werden. Diesen CSS-Ausdruck können Sie dann zum Beispiel mit Facepager[10] ausprobieren. Dazu fügen Sie die URL einer Webseite in Facepager als Startknoten ein, laden die Seite herunter und passen dann mittels CSS-Ausdrücken die angezeigten Spalten an (☛ *Repositorium*).

4.1.3 XPath

Während CSS für die Gestaltung von Webseiten konzipiert wurde, ist XPath direkt für die Adressierung von Daten in XML-verwandten Dokumenten entworfen worden (Abb. 4.4). Auch hier erfolgt die Auswahl von Elementen zunächst über deren

[10] Siehe Jünger und Keyling (2022; https://github.com/strohne/Facepager).

Namen. Ein Dokument wird wiederum als hierarchische Baumstruktur aufgefasst, die mit Achsen wie beispielsweise `child` (untergeordnetes Element), `parent` (übergeordnetes Element) oder `preceding` und `following` (vorangegangenes oder nachfolgendes Element) durchsucht wird. Der folgende Ausdruck würde ausgehend vom `html`-Element über das `body`-Element alle Tabellen `table`, darin die Zeilen `tr` und schließlich die Zellen `td` finden.

```
html/child::body/child::table/child::tr/child::td
```

Die Angabe der `child`-Achse kann im Gegensatz zu anderen Achsen entfallen, sodass der folgende vereinfachte Ausdruck die gleiche Funktion erfüllt:

```
html/body/table/tr/td
```

Das Wurzelelement eines Dokuments wird durch den vorgestellten Schrägstrich angesprochen. So würde der folgende Ausdruck immer noch das gleiche Ergebnis liefern, wenn das Wurzelelement `html` ist:

```
/body/table/tr/td
```

Sollen Elemente in beliebiger Tiefe des Baums angesprochen werden, kann der doppelte Schrägstrich verwendet werden. So werden sämtliche `td`-Elemente gefunden:

```
//td
```

Zusätzliche Bedingungen können in jedem Selektionsschritt in eckigen Klammern angegeben werden. Im einfachsten Fall gibt man die Nummer des Elements an, etwa um das dritte `td`-Element zu finden:

```
//td[3]
```

Sinnvoll kann auch mit dem `@`-Zeichen die Eingrenzung auf Attribute sein:

```
//td[@class == 'data']
```

Da Klassenattribute mehrere Werte enthalten können (siehe das Beispiel in Tab. 4.3), ist die Funktion `contains()` nützlich. Hiermit muss das Attribut nicht identisch mit dem Suchausdruck sein, sondern kann auch weitere Texte enthalten:

```
//td[contains(@class,'data')]
```

Die bisherigen Ausdrücke zielen genauso wie CSS-Selektoren immer auf die Tags oder Elemente ab. Ein Element besteht aus dem Elementnamen, Attributen und kann Text enthalten (siehe Kap. 3). Anders als bei CSS sind auch Textinhalte und Attribute direkt als sogenannte Knoten ansprechbar. Mit XPath können die Attribute also nicht nur zur Auswahl von ganzen Elementen verwendet, sondern auch direkt referenziert werden, sodass ein Element wie `<a href="www.example.org">Beispielseite</a>` nach dem Extrahieren nicht noch weiter zerlegt werden muss, um an die Inhalte zu gelangen. Die folgenden beiden Ausdrücke extrahieren einmal den Textinhalt aller Links und einmal alle Werte der `href`-Attribute, das heißt die Zieladressen von Links:

```
//a/text()
//a/@href
```

XPath unterstützt zudem Funktionen, um die gefundenen Knoten weiterzuverarbeiten. Nützlich ist neben der im letzten Beispiel verwendeten `text()`-Funktion etwa die `string()`-Funktion, um den gesamten Text eines Elements zu extrahieren, selbst wenn darin weitere Elemente verschachtelt sind. Das ist zum Beispiel bei Tabellenzellen der Fall, wenn sie Links oder formatierten Text enthalten. Das Beispiel extrahiert den Text der ersten Tabellenzelle im Dokument:

```
string(//td[1])
```

Die Sprache XPath ist komplexer aufgebaut als CSS (Tab. 4.4), erlaubt dafür aber die Formulierung sehr präziser Bedingungen.[11] Mit diesem Wissen lassen sich zum Beispiel Tabellen aus Wikipedia-Seiten in CSV-Tabellen umwandeln. Probie-

[11] Eine umfangreichere Einführung in XPath finden Sie unter anderem unter Selfhtml (2021; https://wiki.selfhtml.org/wiki/XML/XSL/XPath).

Tab. 4.4 Beispiele für XPath-Ausdrücke

Selektor	Bedeutung	Beispiel
//a	Alle a-Elemente	Real Humans
//.[contains (@class, ‘bold’)]	Alle Elemente mit der Klasse „bold“	Real Humans – Echte Menschen
//. [@id='footer')]	Das Element mit der ID „footer“	<div id="footer">5 Ergebnisse</div>
//a[contains (@class, 'extern')]	Alle a-Elemente mit der Klasse „extern“	Example
//p [@id='teaser')]	Alle p-Elemente mit der ID „teaser“	<p id="teaser">Eine Einleitung leitet ein.</p>"
//p//a	Alle a-Elemente innerhalb von p-Elementen	<p id="teaser">Eine ** Einleitung** leitet ein.</p>
//p/a	Alle a-Elemente, die direkt einem p-Element untergeordnet sind	<p id="teaser">Eine ** Einleitung** leitet ein.</p>

Die Selektoren sind in der Tabelle aus Darstellungsgründen umgebrochen. Sie funktionieren nur ohne Umbrüche und Leerzeichen. Quelle: eigene Darstellung

ren Sie es mit der Importfunktion von Google Spreadsheets[12] oder mit Facepager[13] aus! In der Anwendung Facepager finden Sie auch fertige Presets mit ausformulierten XPath-Ausdrücken, um Tabellen zu extrahieren (☛ *Repositorium*).

4.1.4 SQL

Umfangreiche Datenbestände, die sich über mehrere Tabellen verteilen, liegen häufig in relationalen Datenbanken vor (siehe Kap. 3). Solche Datenbanken werden unter anderem in Content-Management-Systemen eingesetzt, angefangen von kleinen Wordpress-Seiten bis zu großen Projekten wie Wikipedia. Hier besteht die Herausforderung nicht vorrangig darin, die Daten in einer unübersichtlichen Struktur zu identifizieren, sondern aus den gesamten Beständen die gewünschten Daten auszuwählen. Ein Standard für die Abfrage von relationalen Datenbanken ist die Structu-

[12] Siehe Google (2022c; https://spreadsheets.google.com).

[13] Siehe Jünger und Keyling (2022; https://github.com/strohne/Facepager).

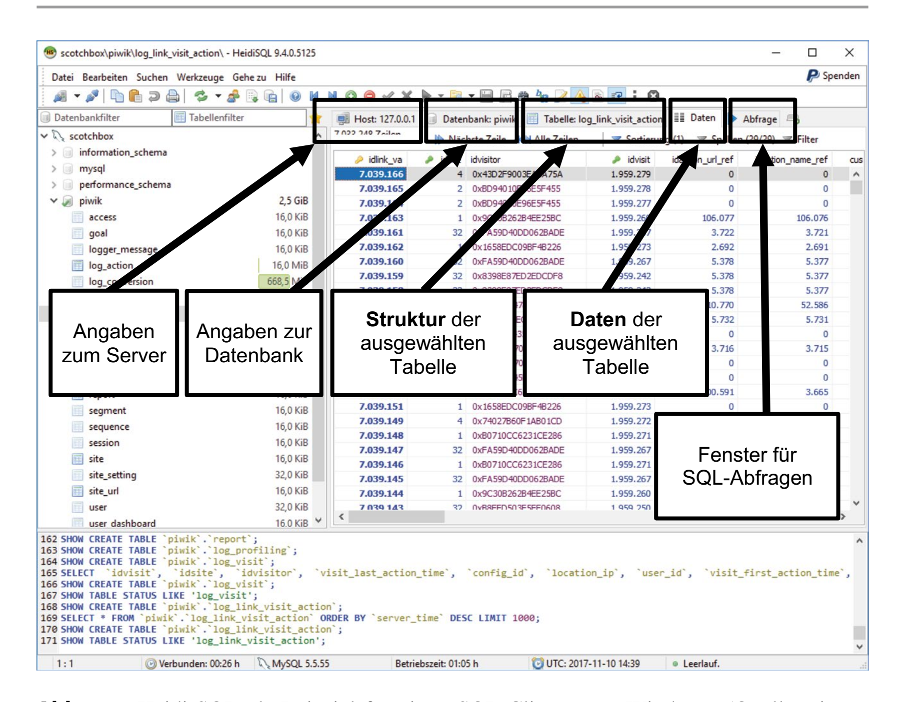


Abb. 4.5 Heidi-SQL als Beispiel für einen SQL-Client unter Windows. (Quelle: eigene Darstellung)

red Query Language (SQL). Dazu werden in einem Datenbank-Client Abfragen formuliert, die vom Datenbankmanagementsystem (DBMS) bearbeitet werden.

Die Datenbanken selbst werden meistens auf Servern installiert. Gängige serverbasierte Datenbankmanagementsysteme im Open-Source-Bereich sind MySQL und MariaDB.[14] Sie können entsprechende Datenbank-Server durchaus auf dem eigenen Computer einrichten, um etwa die Wikipedia-Datenbank zu spiegeln (siehe Abschn. 6.3). Der Zugriff auf die Daten kann dann über vier Wege erfolgen:

1. Über die Kommandozeile wird eine eigene mySQL-Kommandozeile aufgerufen, um darin SQL-Befehle einzugeben.
2. Auf dem Datenbank-Server werden spezielle Weboberflächen wie phpMyAdmin oder Adminer eingerichtet, um SQL-Befehle über den Browser zu verschicken.

[14] Siehe Oracle (2021; https://www.mysql.com/) und MariaDB Foundation (2022; https://mariadb.org/).

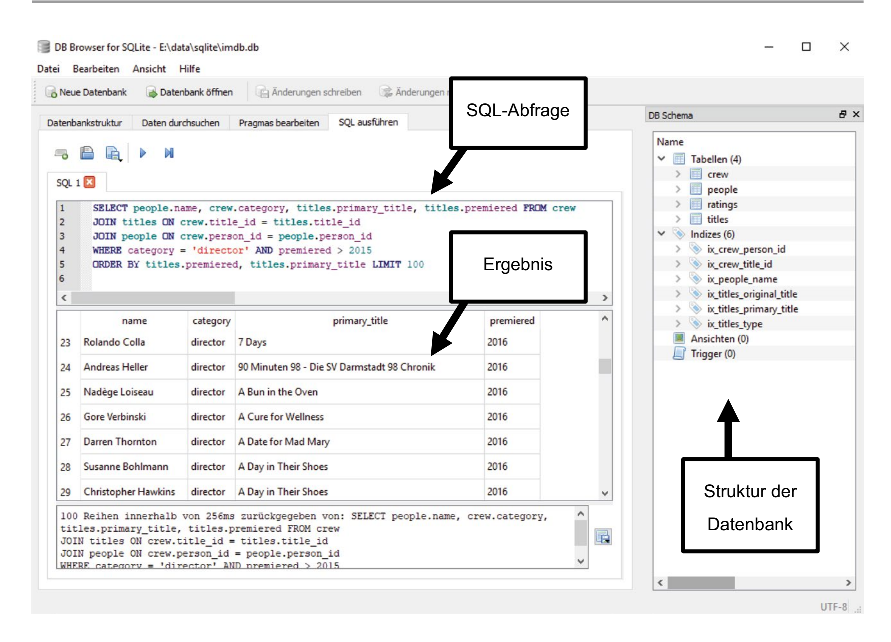


Abb. 4.6 DB Browser for SQLite. (Quelle: eigene Darstellung)

3. Desktop-Anwendungen wie DBWeaver, DataGrip oder HeidiSQL (Abb. 4.5) laufen auf dem eigenen Computer. Auch Entwicklungsumgebungen wie RStudio beinhalten Tools zum Umgang mit Datenbanken.
4. Datenbanken können über R- oder Python-Skripte und mit vielen anderen Programmiersprachen angesprochen werden.

Die verschiedenen Datenbank-Clients ähneln sich meistens und bieten mindestens getrennte Ansichten für die Verwaltung der Tabellen*struktur* und für den Zugriff auf die Tabellen*daten*.

Eine einfache und verbreitete Alternative zu Server-Datenbanken ist SQLite.[15] Diese Datenbanken bestehen aus einer einzigen Datei, die problemlos weitergegeben werden kann und auch keine Server-Installation benötigt. Der Zugriff ist über Programme wie DB Browser for SQLite[16] (Abb. 4.6) möglich, aber auch direkt über R und Python (siehe Kap. 5). Ein Beispiel für eine SQLite-Datenbank finden Sie im ☛ *Repositorium*. Wenn Sie sich DB Browser for SQLite installieren

[15] Siehe SQLite Development Team (2022; https://www.sqlite.org/).

[16] Siehe DB Browser for SQLite (2022; https://sqlitebrowser.org/).

und die Datenbank damit öffnen, können Sie die folgenden SQL-Beispiele direkt nachvollziehen.[17]

Ganz grundsätzlich lassen sich drei Arten von SQL-Befehlen[18] unterscheiden:

1. **Data Definition Statements** werden dazu eingesetzt, die Datenbankstruktur zu erzeugen. Beispielsweise legen Sie mit einem `CREATE DATABASE`-Befehl eine Datenbank an.
2. **Data Manipulation Statements** werden dazu eingesetzt, Daten zu ändern, einzufügen oder zu löschen. Mit `INSERT`-Befehlen werden Daten in eine Datenbank importiert, mit `UPDATE` verändert und mit `DELETE` gelöscht.
3. Eine Unterkategorie der Data Manipulation Statements sind Befehle zur **Abfrage von Daten**. Am wichtigsten ist dafür der `SELECT`-Befehl.

Wenn mehrere Befehle hintereinander ausgeführt werden sollen, dann wird jeder Befehl mit einem Semikolon abgeschlossen. Für die einfache Abfrage von Daten ist das aber nicht nötig. Eine Abfrage aller Spalten und Zeilen einer Tabelle sieht wie folgt aus, wobei `titles` eine Tabelle in der Datenbank bezeichnet:

```
SELECT * FROM titles
```

Diese Abfrage lässt sich um verschiedene Elemente erweitern. So kann statt des Sternchens eine Liste der gewünschten Spalten angegeben werden. Die Angaben `title_id`, `genres` und `primary_title` bezeichnen Spalten in der Tabelle `titles`:

```
SELECT title_id, genres, primary_title FROM titles
```

Die Daten können auch sortiert werden. Im folgenden Beispiel wird nach der Spalte `premiered` (Erscheinungsjahr) sortiert, die Angabe `ASC` (engl. *ascending* = aufsteigend) bewirkt eine aufsteigende Sortierung. Diese Angabe kann weggelassen werden oder durch die Angabe `DESC` (engl. *descending* = absteigend) ersetzt werden:

---

[17] Weitere Möglichkeiten zum Ausprobieren von SQL-Befehlen finden Sie bei W3Schools (2022d; https://www.w3schools.com/sql/) oder Feasel (2021; http://sqlfiddle.com/).

[18] Eine Dokumentation mit allen SQL-Befehlen und der Syntax findet sich in den Handbüchern des jeweiligen DBMS, zum Beispiel im MySQL-Handbuch (Oracle 2022b; https://dev.mysql.com/doc/).

```
SELECT * FROM titles
ORDER BY premiered ASC
```

Außerdem kann die Menge der Daten mit einer `WHERE`-Angabe eingeschränkt werden. Das heißt, es wird danach gefiltert, unter welchen Bedingungen Zeilen ausgewählt werden sollen. Im folgenden Beispiel werden nur Zeilen zurückgegeben, in denen in der Spalte `premiered` der numerische Wert `2019` steht:

```
SELECT * FROM titles
WHERE premiered = 2019
```

Mehrere Kriterien werden mit den booleschen Operatoren `OR`, `AND`, `NOT` und Klammern angegeben. Es können mit dem Ungleichoperator <> anstelle des Gleichheitsoperators = auch alle Zeilen ausgegeben werden, die einen Wert nicht enthalten. Eine Besonderheit stellt der `LIKE`-Operator dar, mit dem Zeichenwerte verglichen werden. Zeichenwerte werden grundsätzlich in Anführungszeichen angegeben. Dabei steht das Prozentzeichen `%` innerhalb der Anführungszeichen als Platzhalter für beliebige Zeichen. Die folgende Anweisung gibt alle Zeilen mit einem Wert größer `2015` in der `premiered`-Spalte zurück, die in der `genres`-Spalte die Zeichenkette `"Western"` enthalten:

```
SELECT * FROM titles
WHERE premiered > 2015
AND genres LIKE '%Western%'
```

Häufig müssen bei der Datenanalyse Daten aus mehreren Tabellen kombiniert werden (siehe Abschn. 4.2). In der Regel gibt es in jeder Tabelle eine Spalte mit einer eindeutigen ID. Diese ID kann dann in einer anderen Tabelle als sogenannter Fremdschlüssel verwendet werden, um die Daten miteinander zu verknüpfen. Soll zum Beispiel in einer Filmdatenbank erfasst werden, welche Schauspieler:innen in welchen Filmen mitgespielt haben, dann werden eine Tabelle mit den Akteuren und eine Tabelle mit den Filmen benötigt. Eine dritte Tabelle verknüpft die IDs der Akteure und der Filme miteinander.

Zur Abfrage verknüpfter Tabellen werden `JOIN`-Anweisungen verwendet (siehe unten Abschn. 4.2.3). Die folgende Anweisung verknüpft die Tabelle `crew` (eine Tabelle mit Filmbesetzungen) mit der Tabelle `titles` (eine Tabelle mit den Filmen). Welche Spalten miteinander abgeglichen werden sollen, steht in der `ON`-Anweisung, die ähnlich wie die `WHERE`-Anweisung funktioniert. Vor dem Spaltennamen muss allerdings der Tabellennamen gefolgt von einem Punkt angegeben

werden, um die Spalten eindeutig zu identifizieren. Diese Zusatzangabe ist nötig, denn es könnten gleichnamige Spalten in verschiedenen Tabellen enthalten sein:

```
SELECT * FROM crew JOIN titles
ON crew.title_id = titles.title_id
```

Bei umfangreichen Datenbeständen kann eine solche Abfrage sehr lange dauern. Um dem zu begegnen, kann die Datenmenge eingeschränkt werden. Die `LIMIT`-Anweisung bewirkt (nicht nur bei `JOIN`), dass nur die angegebene Anzahl an Datensätzen zurückgegeben wird. Dabei ist es sinnvoll, eine Sortierung anzugeben, sodass immer die ersten Datensätze aus der sortierten Liste ausgewählt werden:

```
SELECT * FROM crew JOIN titles
ON crew.title_id = titles.title_id
ORDER BY titles.premiered LIMIT 100
```

Außerdem kann die Datenmenge mit einer `WHERE`-Anweisung weiter eingegrenzt werden. Es werden immer zuerst Joins definiert, dann Where-Einschränkungen vorgenommen, dann folgt die Angabe der Sortierung und schließlich die Limitierung, diese Reihenfolge muss eingehalten werden:

```
SELECT * FROM crew JOIN titles
ON crew.title_id = titles.title_id
WHERE category = 'director'
ORDER BY titles.premiered LIMIT 100
```

In einer Abfrage können mehrere Joins, Sortierungen und Where-Kriterien verwendet werden. Die folgende Abfrage gibt die Regisseur:innen von Filmen aus der IMDb zurück, indem die Crew-Tabelle sowohl mit der Filmtitel- als auch der Personentabelle verknüpft wird:

```
SELECT * FROM crew
JOIN titles ON crew.title_id = titles.title_id
JOIN people ON crew.person_id = people.person_id

WHERE category = 'director' AND premiered > 2000
ORDER BY titles.premiered, titles.primary_title
LIMIT 100
```

Zur Berechnung des Ergebnisses einer solchen Abfrage muss das DBMS mehrere Spalten auswerten, bevor das Ergebnis zurückgegeben werden kann. Das betrifft vor allem die Spalten in der `ON`-, der `WHERE`- und der `ORDER BY`-Anweisung. Diese Auswertung kann sehr ressourcenintensiv sein und entsprechend lange dauern. Um die Zugriffszeit zu verkürzen, werden deshalb sogenannte Indizes angelegt. Ein solcher Index enthält eine Art Adressbuch, das heißt, die Daten werden so vorsortiert, dass über den Wert einer Spalte schnell auf die anderen Inhalte der Tabelle zugegriffen werden kann. Er wird aus einer oder mehreren Spalten einer Tabelle gebildet und kann über den Befehl `CREATE INDEX` erstellt werden. Das Aktualisieren der Indizes erhöht zwar den Aufwand beim Verändern der Daten, meistens sind Lesezugriffe in einer SQL-Datenbank aber häufiger als Schreibzugriffe. Immer wenn eine Abfrage also sehr lange braucht, sollten Sie mit Ihrem Datenbank-Client die Indizes überprüfen und ggf. neue Indizes anlegen.

Übungsfragen

1. Was bedeutet der reguläre Ausdruck `[0-9]+` ?
2. Welche Quantoren gibt es in regulären Ausdrücken?
3. Wie kann ein Wort am Anfang einer Zeile gefunden werden?
4. Wofür stehen die Zeichen `\r` und `\n`?
5. Öffnen Sie im ☛ *Repositorium* die Datei *example_text.txt* mit einem Texteditor und suchen Sie mit regulären Ausdrücken nach allen a) Jahreszahlen, b) Prozentzahlen und c) geschützten Leerzeichen!
6. Welche Schwierigkeiten ergeben sich bei der Datenextraktion in Bezug auf Sonderzeichen?
7. Wie können in einem HTML-Quelltext die URLs gefunden werden?
8. Mit welchem CSS-Selektor und mit welchem XPath-Ausdruck können alle Zeilen einer HTML-Tabelle gefunden werden?
9. Was sind eine Datenbank, ein Datenbank-Management-System und ein Datenbank-Client?
10. Formulieren Sie einen SQL-Befehl, mit dem die Felder `titel` und `jahr` aus einer Tabelle `movies` abgefragt werden. Grenzen Sie auf Titel aus den Jahren 2005 bis 2010 ein!

Referenzen

Çelik, T., Etemad, E. J., Glazman, D., Hickson, I., Linss, P. & Williams, J. (2018). *Selectors Level 3. W3C Recommendation 06 November 2018.* Zugriff am 27.01.2022. https://www.w3.org/TR/selectors-3/

Hazel, P. (2021). *Perl-compatible Regular Expressions (PCRE)*. Zugriff am 27.01.2022. https://www.pcre.org/original/doc/html/index.html

Oracle. (2022). *MySQL 8.0 Reference Manual. Chapter 13 SQL Statements*. Zugriff am 27.01.2022. https://dev.mysql.com/doc/refman/8.0/en/sql-statements.html

Robie, J., Dyck, M. & Spiegel, J. (2017a). *XML Path Language (XPath) 3.1. W3C Recommendation 21 March 2017*. Zugriff am 27.01.2022. https://www.w3.org/TR/2017/REC-xpath-31-20170321/

Robie, J., Dyck, M. & Spiegel, J. (2017b). *XQuery 3.1: An XML Query Language. W3C Recommendation 21 March 2017*. Zugriff am 27.01.2022. https://www.w3.org/TR/xquery-31/

4.2 Transformationsverfahren

Daten liegen selten in der Form vor, die für eine Analyse notwendig ist. Vor allem bei der Arbeit mit Textdaten besteht ein wesentlicher Schritt darin, unstrukturierte Daten zu strukturieren. In der Regel strebt man dabei an, eine oder mehrere Tabellen zu erstellen. Dafür haben sich einige hilfreiche Regeln etabliert (Wickham 2014, S. 4):[19]

1. Jede **Beobachtungseinheit** wird in einer eigenen **Tabelle** abgespeichert. Wenn es beispielsweise um Webseiten, Nutzer:innen und Posts geht, wird für jeden dieser Typen in der Regel eine eigene Tabelle verwendet.
2. Jede **Beobachtung** entspricht einer eigenen **Zeile**. Beobachtungen oder Fälle sind etwa die Posts. Die Eigenschaften und Inhalte eines Posts befinden sich möglichst alle in einer einzigen Zeile.
3. Jede **Variable** wird in einer eigenen **Spalte** vorgehalten. Eine Variable wäre zum Beispiel die Anzahl der Kommentare eines Posts. Im Kopf der Tabelle steht dann der Name der Variable und die Werte werden darunter aufgeführt.
4. Jeder **Wert** ist in einer eigenen **Zelle** abgelegt. Eine Zelle ist dabei der Ort, an dem sich Spalte (vertikal) und Zeile (horizontal) treffen. Die Anzahl der Kommentare eines jeden Posts liegt also in einzelnen Zellen, es sollten möglichst nicht mehrere Angaben in einer gemeinsamen Zelle liegen.

Wer viel Datenanalyse betreibt, für den werden diese Regeln nach und nach selbstverständlich. Es geht darum, die Welt aus einer rechteckigen Brille zu betrachten,

[19] Datenstrukturen, die diesen Regeln entsprechen, werden in der Programmiersprache R auch Tibbles genannt.

weil diese Perspektive übersichtlich ist. Vor allem für die Zusammenarbeit mit anderen ist es sehr hilfreich, sich an diese Konventionen zu halten. Denn dadurch lassen sich auch fremde Daten schnell verstehen. Das gilt auch, wenn Sie mit Excel oder ähnlichen Programmen arbeiten: Die erste Zeile sollte für Variablennamen reserviert bleiben, alle anderen Zeilen sind für die Fälle da. Verbundene Zellen sollten vermieden werden. Anmerkungen sollten nicht irgendwo in die Zellen geschrieben werden, dafür lässt sich die Anmerkungsfunktion verwenden oder Anmerkungen werden in einer eigenen Spalte oder einem eigenen Tabellenblatt erfasst. So wird sichergestellt, dass Dateien für beliebige Verarbeitungsprogramme kompatibel sowie dokumentiert und damit auch für andere intuitiv lesbar sind.

4.2.1 Umformen von Tabellen: Wide- und Long-Format

Die rechteckige Brille reduziert nicht nur Komplexität, sondern erlaubt auch einen vielseitigen Blick auf Daten, mitunter muss man dazu jedoch die Brille drehen. Stellt man sich einen Datensatz mit Posts auf einer Social-Media-Plattform vor, in der jede Zeile ein Post enthält und in den Spalten Eigenschaften wie die Anzahl der Likes oder Kommentare angegeben sind (Tab. 4.5), kann dieser Datensatz in verschiedene Richtungen gedreht werden. Wenn man nicht an der Analyse von Posts,

Tab. 4.5 Wide-Format

post_id	likes	comments	shares
1	0	6	2
2	20	0	8
3	6	0	0
4	5	0	0
5	0	4	1
6	64	0	1

Quelle: Eigene Darstellung

Tab. 4.6 Long-Format

post_id	variable	value
1	likes	0
2	likes	20
3	likes	6
4	likes	5
5	likes	0

Fortsetzung

Tab. 4.6 (Fortsetzung)

post_id	variable	value
6	likes	64
1	comments	6
2	comments	0
3	comments	0
4	comments	0
5	comments	4
6	comments	0
1	shares	2
2	shares	8
3	shares	0
4	shares	0
5	shares	1
6	shares	1

Quelle: Eigene Darstellung

Tab. 4.7 Beispiel mit mehreren Hashtags in einer Zelle

id	text	hashtags
1	#Lastenräder-Projekt „CoBiUM-Cargo bikes in urban mobility“ von #UniGreifswald nimmt Fahrt auf…	Lastenräder;UniGreifswald
2	Mit F.U.N. in die Wildnis! #citizenscience an der Uni Greifswald: Fledermauslehrpfad im #Naturpark…	citizenscience;Naturpark
3	#Nachhaltigkeit war ein großes #Thema beim Unique-Ideenwettbewerb 2019! 7 Teams stellten ihre #Ideen der Fachjury vor …	Nachhaltigkeit;Thema;Ideen
4	machine-learning für die #Ökologie? Das #Projekt DIG-IT! von @wissen_lockt & @Fraunhofer_IGD widmet sich der Auswertung …	Ökologie;Projekt
5	Das ist an der #unigreifswald #wissenlocktmich 😃😃 …	unigreifswald;wissenlocktmich

Quelle: Eigene Darstellung in Anlehnung an den Twitterkanal der Universität Greifswald

sondern an den Reaktionen auf die Posts interessiert ist, kann sich eine Umformung der Tabelle lohnen: Aus den Variablen (Likes oder Kommentare) werden dann Fälle. Man unterscheidet dabei zwei grundlegende Formate.

Im **Wide-Format** (Tab. 4.5) ist für jede Eigenschaft oder Variable eine Spalte vorgesehen. So gibt es etwa Spalten mit den Anzahlen der Likes, Kommentare und

Shares eines Posts. Je mehr Eigenschaften in einem Datensatz gespeichert werden, desto breiter wird die Tabelle.

Im **Long-Format** (Tab. 4.6) reichen für die gleichen Daten drei Spalten aus. In einer Spalte wird der Name einer Eigenschaft oder Variablen festgehalten und in einer zweiten Spalte der Wert. Damit klar ist, welche Eigenschaften zusammengehören, wird für jeden Fall eine Nummer oder ein anderer eindeutiger Bezeichner verwendet (`post_id`). Je mehr Eigenschaften in einem Datensatz gespeichert werden, desto länger wird die Tabelle.

Außerdem kommt es vor, dass mehrere Werte in einer Zeile zusammengefasst sind. Man spricht auch von verschachtelten (engl. *nested*) Daten. Wenn man beispielsweise mit Facepager Tweets erhebt (siehe Abschn. 7.2), kann daraus eine Liste von Hashtags, die durch Semikola getrennt sind, erzeugt werden (Tab. 4.7). Um die Hashtags auswerten zu können, müssen sie aufgetrennt werden, sodass in jeder Zeile ein einzelnes Hashtag steht. Dieses Verfahren nennt sich Unnesting (deutsch *entschachteln*). Wenn Text in einzelne Teile zerlegt wird, etwa in die Wörter eines Satzes, spricht man auch von Tokenisierung. Auf diese Weise, wenn jedes Token (= Wort oder Hashtag) in einer eigenen Zeile steht, lässt sich leicht auszählen, welche Wörter oder Hashtags besonders häufig vorkommen.

Tab. 4.8 Befehle zum Umformen von Tabellen (Beispiele)

Programm/ Sprache	**von wide zu long**	**von long zu wide**

R (*tidyverse*)	`pivot_longer()`
	`gather()`
	`melt()`
		`pivot_wider()`
		`spread()`
		`cast()`

Python (*pandas*)	`stack()` `melt()`	`unstack()` `pivot()`
Excel	manuell untereinander kopieren	PIVOT-Funktion

Programm/ Sprache	**Richtung**	**nested zu unnested**	**von unnested zu nested**
R (*tidyverse*)	vertikal	`separate_rows()` `unnest_token()`	`group_by()` und `paste0()` mit dem parameter `collapse`
	horizontal	`separate()`	`unite()`
Python (*pandas*)	vertikal	`str.split()` und `explode()`	`groupby()` und `join()`
Excel	horizontal	Funktion TEXT IN SPALTEN	`&`-Formel

Quelle: Eigene Darstellung

Statistikprogramme und die entsprechenden Programmiersprachen stellen in der Regel Funktionen zur Umformung zwischen Wide- und Long-, Nested- und Unnested-Formaten zur Verfügung (Tab. 4.8). In R erstellt zum Beispiel die Funktion `separate_rows()` für jedes Token eine neue Zeile, es wird also vertikal entschachtelt. Dagegen kann man mit `separate()` horizontal entschachteln, das heißt, eine Spalte wird in mehrere neue Spalten aufgeteilt. Umgekehrt lassen sich auch mehrere Spalten zu einer zusammenfassen. Beim Verschachteln und Entschachteln muss man sich also entscheiden, ob eine Tabelle im Long- oder im Wide-Format entstehen soll.

Die konkrete Umsetzung hängt vom Anwendungsfall und von der verwendeten Software ab (☛ *Repositorium*). Wichtig ist zunächst, die grundlegenden Konzepte zu verstehen. Denn die Umformung von Datensätzen kann die Arbeit wesentlich erleichtern und erweitert die Möglichkeiten der Datenanalyse.

4.2.2 Zusammenführen von Daten: Joins

Viele Fragestellungen erfordern, dass mehrere Datensätze aus unterschiedlichen Quellen zusammengeführt oder miteinander abgeglichen werden. Im einfachsten Fall haben die verschiedenen Datensätze die gleiche Form und können untereinander oder nebeneinander kopiert werden. Mit R können dafür etwa die *tidyverse*-Funktionen `bind_rows()` oder `bind_cols()` verwendet werden. In Python wird dafür die *pandas*-Funktion `concat()` verwendet, mit dem Achsenparameter wird gesteuert, ob Spalten (`axis=1`) oder Zeilen (`axis=0`) aneinandergebunden werden sollen (siehe Kap. 5).

Häufig sind jedoch nicht alle benötigten Daten bereits in gleichartigen Tabellen vorhanden. Angenommen: In einer Tabelle sind Zeitschriften erfasst und es sollen die Webseiten dieser Zeitschriften analysiert werden – es fehlen aber die Webadressen. Die Webadressen (URLs) liegen in einer zweiten Tabelle, wobei die beiden Tabellen nicht deckungsgleich sind. In beiden Tabellen finden sich Zeitschriften, die in der jeweils anderen fehlen. Im besten Fall liegen aber ein oder mehrere Merkmale vor, die zum Abgleich verwendet werden können, zum Beispiel eine International Standard Serial Number (ISSN).

In Abb. 4.7 sind zwei Tabellen dargestellt, die sich nicht vollständig überschneiden. Die beiden orange markierten ISSN kommen nicht in der jeweils anderen Tabelle vor. Die grün markierte ISSN kommt beispielsweise in beiden Tabellen vor.

Beim Zusammenführen der Tabellen ist zu entscheiden:

- Sollen nur die übereinstimmenden Zeilen erhalten bleiben, wird ein sogenannter **Inner Join** verwendet.

- Sollen alle Zeilen in beiden Tabellen erhalten bleiben, dann kann ein **Full Join** eingesetzt werden. In den Zeilen ohne Übereinstimmung bleiben die Felder leer.
- Wenn im Ergebnis nur die erste Tabelle wichtig ist, kann ein **Left Join** eingesetzt werden. Hierbei bleiben alle Zeilen der linken Tabelle erhalten und überall wo es möglich ist, werden die Zeilen aus der rechten Tabelle zugeordnet. Das Gegenstück ist der **Right Join**, der nach dem gleichen Prinzip funktioniert. Hier bleiben alle Datensätze aus der rechten Tabelle erhalten.

Aufpassen muss man, wenn das Vergleichskriterium mehrfach vorkommt, wenn also zum Beispiel eine Zeitschrift doppelt aufgeführt wurde oder eine ISSN versehentlich doppelt vergeben ist. Dann werden auch im Ergebnis die Zeilen vervielfältigt. Das Zusammenführen kann man sich so vorstellen, dass zunächst *alle* Kombinationen aus allen Zeilen in den beiden Datensätzen gebildet werden (= Kreuzprodukt). Im nächsten Schritt werden dann diejenigen Zeilen wieder aussortiert, die der Join-Bedingung *nicht* entsprechen.

Die Tabellen werden in der Regel so zusammengeführt, dass im Ergebnis alle Spalten aus den ursprünglichen Tabellen vorhanden sind. Einige Programmiersprachen unterstützen Join-Varianten, bei denen nur die Spalten der ersten Tabelle erhalten bleiben:

Tabelle mit Zeitschriften

zeitschrift	issn
Cognitive Linguistics	0936-5907
Educational Studies	0013-1946
Journal of Personality …	0022-3514
Journal of Popular Film …	0195-6051
Mass Communication …	1016-1007
Negotiation	1546-9522
Pennsylvania Com…	2372-6350
Popular Communication	1540-5702
South African Journal of …	0257-2117
Studies in European …	1741-1548

Tabelle mit URLs

issn	url
0013-1946	www.heds.com
0021-3624	www.mjei.com
0195-6051	www.vjpf.com
0257-2117	www.rjal.com
0737-7363	www.ujch.com
0883-2323	www.vjeb.com
0967-2567	www.rejh.com
1472-9342	www.rouc.com
1540-5702	www.hppc.com
1741-1548	www.rseu.com

Abb. 4.7 Zwei Tabellen vor dem Zusammenführen. Quelle: Eigene Darstellung in Anlehnung an Taylor & Francis (2022)

- Mit einem **Semi Join** bleiben in der ersten Tabelle nur die Zeilen erhalten, für die ein Eintrag in der zweiten Tabelle gefunden werden konnte. Das entspricht im Prinzip einem Left Join, es werden aber keine Daten aus der zweiten Tabelle übernommen.
- Bei einem **Anti Join** bleiben in der ersten Tabelle nur die Zeilen erhalten, für die *kein* Eintrag in der zweiten Tabelle gefunden werden konnte. So können Datensätze identifiziert werden, die in der zweiten Tabelle fehlen.

In der Programmiersprache R (siehe Abschn. 5.1) werden die genannten Verfahren unter anderem über Bibliotheken aus dem Tidyverse umgesetzt.[20] Das folgende R-Skript vollzieht einen Left Join, wobei `journal` und `url` jeweils den in Abb. 4.7 dargestellten Datensatz enthalten. Das Kriterium zum Abgleichen wird über den `by`-Parameter angegeben und das Ergebnis in `x` abgelegt.

```
x <- left_join(journal, urls, by = "issn")
```

Daraus entsteht die Tabelle in Abb. 4.8; nur bei den passenden Zeilen konnte die URL ergänzt werden, in allen anderen Fällen bleibt die neue Spalte leer. Es können

zeitschrift	issn	url
Cognitive Linguistics	0936-5907	
Educational Studies	0013-1946	www.heds.com
Journal of Personality …	0022-3514	
Journal of Popular Film …	0195-6051	www.vjpf.com
Mass Communication …	1016-1007	
Negotiation	1546-9522	
Pennsylvania Com…	2372-6350	
Popular Communication	1540-5702	www.hppc.com
South African Journal of …	0257-2117	www.rjal.com
Studies in European …	1741-1548	www.rseu.com

Abb. 4.8 Mit einem Left Join zusammengeführte Tabelle. Quelle: Eigene Darstellung in Anlehnung an Taylor & Francis (2022)

[20] Für komplexere Verfahren lohnt ein Blick in das Package *fuzzyjoin* (Robinson et al. 2020).

auch mehrere Kriterien gleichzeitig zum Abgleich verwendet werden. Würden die Ausgangstabellen eine weitere Spalte mit einer Jahresangabe enthalten und soll diese Angabe zum Abgleich verwendet werden, kann über den `by`-Parameter eine Liste `c()` mit mehreren Spaltennamen angegeben werden:

```
x <- left_join(
  journal, downloads,
  by = c("issn", "jahr")
)
```

Bislang wurde vorausgesetzt, dass die Spaltennamen in beiden Tabellen übereinstimmen. In diesem Fall könnte der `by`-Parameter sogar entfallen, R würde automatisch mit den übereinstimmenden Spaltennamen arbeiten. Wenn sich die Namen unterscheiden, kann ein benannter Vektor verwendet werden. Links vom Gleichheitszeichen steht dann der Spaltenname des linken Datensatzes (hier: `name`) und rechts der Name im rechten Datensatz (hier: `titel`). Nach dem Zusammenführen bleibt in der Regel nur eine Spalte übrig, denn die beiden ursprünglichen Spalten enthalten die gleiche Information:

```
x <- left_join(
  journal, urls,
  by = c("name" = "titel")
)
```

Die meisten Statistik- und Datenbankprogramme sehen Befehle zum Zusammenführen von Datensätzen vor (Tab. 4.9). Besonders umständlich ist das Zusammenführen mit Excel. Hier müssen die Werte Spalte für Spalte und Zeile für Zeile mit der Funktion `SVERWEIS` (engl. `VLOOKUP`) aus der zweiten Tabelle geholt werden. Hilfreich ist bei der Arbeit mit Excel auch die Funktion `ZÄHLENWENN`, mit der festgestellt werden kann, wie viele Datensätze in einem anderen

Tab. 4.9 Beispiele für Left Joins

Sprache	Beispiel für einen Left Join
SQL	`SELECT * FROM journal LEFT JOIN urls ON journal.name = urls.titel`
R (*tidyverse*)	`left_join(journal, urls, by = c("name" = "titel"))`
Python (*pandas*)	`pd.merge(journal, urls, on_left='name', on_right='titel', how='inner')`
Excel	`=SVERWEIS(C2;urls!A:B;2;FALSCH)`

Tabellenblatt enthalten sind. Das folgende Beispiel setzt voraus, dass im Tabellenblatt `zeitschriften` in der Spalte `C` und im Tabellenblatt `urls` in der Spalte `A` das übereinstimmende Merkmal angegeben ist, etwa eine ISSN. Es wird dann berechnet, wie häufig der Wert in `zeitschriften!C2` im Bereich `urls!A:A` vorhanden ist:

```
=ZÄHLENWENN(urls!A:A;zeitschriften!C2)
```

Mit etwas Kreativität und Übung lassen sich damit in Kombination mit den Filter- und Sortierfunktionen auch in Excel Datensätze zusammenführen.

4.2.3 Aggregieren von Daten: Map-Reduce und Split-Apply-Combine

Das Gegenstück zum Zusammenführen von Daten ist die Aggregation, bei der die Daten nicht mehr, sondern weniger werden. Dieser Vorgang ist ganz wesentlich für die Datenanalyse, um Komplexität auf wenige interessante Muster einzuschmelzen. Ein Beispiel dafür ist die Berechnung von Mittelwerten. Beschäftigt man sich mit der Anzahl von Zeitungsberichten zur Coronapandemie, mit der Anzahl der Logins auf einer Webseite oder mit der Anzahl von Terroranschlägen im Zeitverlauf, kann die Anzahl der Einzelaktivitäten zunächst für jeden Tag zusammengezählt werden. Um nun einen typischen Tag zu beschreiben, kann als ein einfaches Maß der Mittelwert aller Tageswerte gebildet werden.[21] Dabei ist man möglicherweise nicht nur daran interessiert, alle Tage gleich zu erfassen, sondern etwa Wochentage vom Wochenende zu unterscheiden.

Hinter einer solchen Datenaufbereitung stehen konzeptionell vier Vorgänge:

- **Map:** Einzelne Werte müssen zunächst Zeile für Zeile transformiert werden. Beispielsweise wird aus dem Zeitpunkt des Logins (2023-08-09T13,00:12.234) eine neue Spalte mit dem Tag des Logins gebildet, das heißt, die Uhrzeit wird

[21] Mittelwerte sind nicht immer eine gute Wahl zur Beschreibung von Verteilungen. In den genannten Beispielen ist mit rechtsschiefen Verteilungen zu rechnen: Einige wenige Monate umfassen sehr viel Aktivität, die meisten aber eher wenig (ein sogenannter Long Tail). Oder auch: Einige Webseiten werden von sehr vielen Menschen genutzt, die meisten aber von sehr wenigen. Hier bietet sich alternativ der Median zur Beschreibung an. Wichtig ist an dieser Stelle, das Aggregationsverfahren passend zur Fragestellung und zur Datenlage zu wählen.

abgeschnitten (2023-08-09) oder der Wochentag wird bestimmt (Mittwoch). Die Loginzeit wird also auf einen Tag abgebildet (engl. *mapped*).
- **Split:** Der gesamte Datensatz wird für den folgenden Schritt in Teile zerlegt, zum Beispiel in sieben Teile für jeden einzelnen Wochentag.
- **Reduce:** Jeder Teil wird zusammengefasst, indem ein neuer Kennwert wie der Mittelwert berechnet wird. Mitunter wird dieser Schritt auch als Apply bezeichnet.
- **Combine:** Die Einzelergebnisse werden wieder zu einer Tabelle zusammengefasst, sodass in einer Spalte der Wochentag und in einer anderen Spalte die mittlere Anzahl der Logins stehen.

In verschiedenen Programmiersprachen und Tools werden diese Vorgänge unterschiedlich bezeichnet oder sind teilweise auch nicht explizit benannt. Unter R findet sich der Vorgang des Abbildens unter anderem in der Tidyverse-Funktion `map()` wieder, das Zerlegen ist über Funktionen wie `group_by()` umgesetzt und `summarize()` reduziert und kombiniert die Werte innerhalb der Gruppen. Im Python-Package *pandas* finden sich äquivalent dazu die Funktionen `map()`, `groupby()` und `aggregate()`.

Unabhängig von der konkreten Programmiersprache ist eine Zergliederung von Problemen in diese vier Vorgänge immer dann besonders hilfreich, wenn umfangreiche Daten verarbeitet werden müssen. Denn selbst wenn Map, Split, Reduce und Combine hintereinander ablaufen, lässt sich im besten Fall innerhalb dieser Schritte parallelisieren. Das bedeutet, die Bearbeitung jeder einzelnen Zeile (map) oder Gruppe (reduce) kann auf unterschiedlichen Computern oder innerhalb eines Computers auf unterschiedlichen Kernen (siehe Abschn. 6.3) ausgeführt werden. Nur auf diese Weise können Suchmaschinen wie Google nahezu das gesamte Web für die Suche aufbereiten. Eine Software dafür ist beispielsweise Apache Hadoop,[22] in der zunächst Funktionen für das Mapping und das Reducing entwickelt werden, die sodann auf einem Computercluster verteilt ausgeführt werden.

4.2.4 Umformen von Matrizen: Ähnlichkeitsberechnungen

Für eine möglichst effiziente Verarbeitung von Daten werden Tabellen als Matrix aufgefasst. Eine Matrix setzt sich aus Vektoren (Zeilen oder Spalten) zusammen und Vektoren beinhalten wiederum eine Sammlung von Elementen (Werten). Je nach Perspektive besteht eine Matrix somit aus einer Liste mit Zeilen oder aus

[22] Siehe Apache Software Foundation (2022a; https://hadoop.apache.org/).

post_id	likes	comments	shares
1	0	6	2
2	20	0	8
3	6	0	0
4	5	0	0
5	0	4	1
6	64	0	1

$$A = \begin{bmatrix} 0 & 6 & 2 \\ 20 & 0 & 8 \\ 6 & 0 & 0 \\ 5 & 0 & 0 \\ 0 & 4 & 1 \\ 64 & 0 & 1 \end{bmatrix}$$

Abb. 4.9 Darstellung der Reaktionen auf Social-Media-Posts als Tabelle (links) und Matrix (rechts). (Quelle: eigene Darstellung)

einer Liste mit Spalten (siehe Abschn. 3.1).[23] Eine Matrix m x n besteht aus m Zeilen und n Spalten, die Matrix in Abb. 4.9 ist somit eine 6 x 3 Matrix. Mathematisch werden Matrizen in eckigen Klammern dargestellt und üblicherweise mit Großbuchstaben benannt, wohingegen Vektoren an Kleinbuchstaben erkennbar sind.

In einer Tabelle lassen sich einzelne Elemente über die Zeilen- oder Spalten*bezeichnungen* adressieren. In Matrizen werden dagegen sogenannte Indizes verwendet – das sind die Zeilen- und Spalten*nummern*. Wenn im Beispiel die Like-Anzahl des Posts mit der ID 4 ausgelesen werden soll, findet sich dieser Wert in der ersten Spalte der vierten Zeile der Matrix. Diese Zelle lässt sich über Indizes adressieren, wobei der Zeilenindex meistens als erstes angegeben wird. Je nach Programmiersprache beginnt die Zählung des Index bei 0 oder bei 1 wie in folgenden beiden Ausdrücken ersichtlich ist:

- Adressierung in R: `A[4,1]`
- Adressierung in Python: `A[3,0]`

Wenngleich die Betrachtung von Tabellen als Matrizen zunächst weniger intuitiv und etwas sperrig erscheinen mag, lohnt sich die Auseinandersetzung damit und das Denken in Matrizen. So können einige Programme Matrizen effizienter verarbeiten, wenn die Funktionen vektorisiert sind. Das heißt, diese Funktionen bearbeiten einen ganzen Vektor, also alle Einträge in einer Zeile oder Spalte, gleichzeitig und man muss keine Schleife programmieren, mit der jeder Eintrag Zeile für Zeile und Spalte

[23] Eine einzelne Liste wird auch als Array bezeichnet und eine Liste mit Listen als mehrdimensionales Array. Das kann weiter fortgesetzt werden, um beispielsweise ein dreidimensionales Array zu erhalten: Ein Bild besteht aus Pixeln, die in Zeilen (erste Dimension) und Spalten (zweite Dimension) angeordnet sind. Jedes Pixel enthält ein Array (dritte Dimension) mit den jeweiligen Werten für den Rot-, Grün- und Blauanteil.

für Spalte durchlaufen wird. Matrizen spielen deshalb in vielen Algorithmen eine wichtige Rolle, insbesondere beim Machine Learning (siehe Kap. 8).

Nicht nur Zahlen, sondern auch Texte lassen sich in Matrizen erfassen: In einer Dokument-Term-Matrix wird jedes Dokument (z. B. ein Post) in einer Zeile erfasst und für jedes Wort wird eine Spalte angelegt. In den Zellen wird dann angegeben, wie häufig dieses Wort im Dokument auftritt (siehe Abschn. 9.1). Vergleicht man die Zeilenvektoren von zwei Dokumenten, so lässt sich daraus die Ähnlichkeit von Texten berechnen: In ähnlichen Texten treten die gleichen Wörter ungefähr gleich häufig auf. Ebenso können Netzwerke als Matrizen erfasst werden, wobei die Zeilen und Spalten dann Personen darstellen. In den Zellen wird zum Beispiel mit einer 1 angegeben, wenn sich zwei Personen kennen (siehe Kap. 10). Auch hier lassen sich Erkenntnisse aus dem Vergleich von Vektoren ziehen. Die Zeilen oder Spalten von zwei Personen sind sich ähnlich, wenn sie die gleichen Bekanntenkreise haben. Auf die gleiche Weise funktionieren kollaborative Empfehlungssysteme, bei denen ausgewertet wird, welche Filme Personen mit einer ähnlichen Nutzungshistorie gesehen, welche Social-Media-Posts ähnliche Nutzer:innen gelikt oder welche Produkte Personen mit ähnlichen Interessen gekauft haben (Ricci et al. 2015).

Text- oder Netzwerkmatrizen zeichnen sich meistens dadurch aus, dass sie viele Zeilen und Spalten enthalten, aber dass die meisten Zellen leer sind (bzw. den Wert 0 enthalten). Wenn man etwa viele unterschiedliche Texte miteinander vergleicht, kommen in jedem Text je nach Thema sehr spezifische Wörter vor. Geht es in einem Social-Media-Post um Kochrezepte und in einem anderen um Volleyball, dann braucht es für jedes Wort[24] in den verschiedenen Vokabularen eigene Spalten, ohne dass alle Texte alle Spalten nutzen. Man unterscheidet deshalb zwei Arten von Matrizen:

- Matrizen sind **dense** (deutsch *dichtbesetzt*), wenn die meisten Zellen Werte enthalten, zum Beispiel, wenn alle Entfernungen zwischen den Großstädten der Welt erfasst werden. Diese Matrizen werden vollständig im Arbeitsspeicher des Computers vorgehalten und können dann schnell über die Indizes angesprochen werden. Beim Abspeichern werden einfach alle Werte in einer langen Reihe abgespeichert und zusätzlich die Information angegeben, an welchen Stellen diese lange Reihe in Zeilen oder Spalten umgebrochen werden muss.[25]

[24] Für jeden *type*, nicht für jedes *token*, siehe Kap. 9.

[25] Matrizen können schnell in andere Dimensionen umgeformt werden, wenn alle Werte hintereinander in einen Vektor gelegt und dann anders umgebrochen werden.

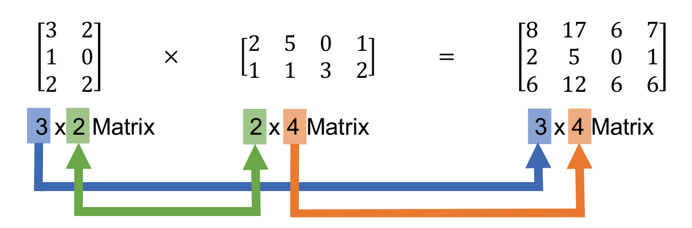


Abb. 4.10 Das Resultat einer Matrixmultiplikation. Die Farben kennzeichnen, welche Dimensionen der Spalten bzw. Zeilen übereinstimmen müssen. (Quelle: eigene Darstellung)

- Matrizen sind **sparse** (deutsch *dünnbesetzt*), wenn die meisten Zellen leer sind. In diesem Fall kann beim Speichern einer Matrix Platz eingespart werden, indem in speziellen Datenformaten nur die tatsächlich vorhandenen Werte abgelegt werden.[26] Insbesondere wenn Millionen von Texten verglichen werden, stößt das andernfalls, ohne eine sinnvolle Komprimierung und optimierte Zugriffsverfahren, schnell an die Grenzen eines einzelnen Computers.

Für beide Varianten werden in den verschiedenen Programmiersprachen entsprechend angepasste Funktionen eingesetzt. Eine besonders häufig benötigte Rechenoperation, die unter anderem zur Berechnung von Textähnlichkeiten verwendet wird, ist die Matrixmultiplikation, bei der eine Matrix mit einer anderen Matrix multipliziert wird. Damit zwei Matrizen miteinander multipliziert werden können, muss eine grundlegende Voraussetzung erfüllt sein: Die Anzahl der Spalten des Multiplikators A (vor dem Mal-Zeichen) muss der Anzahl der Zeilen des Multiplikanden B (nach dem Mal-Zeichen) entsprechen. Das Produkt ist eine dritte Matrix C, die m Zeilen der ersten Matrix A und n Spalten der zweiten Matrix B aufweist (Abb. 4.10).

Konkret wird das Produkt ermittelt, indem zunächst die *erste Zeile* der einen Matrix Element für Element mit der *ersten Spalte* der anderen Matrix multipliziert wird. Die Zwischenergebnisse werden dann addiert und das ergibt den Wert in der ersten Spalte der ersten Zeile der neu errechneten Matrix. Anschließend wird die *erste Zeile* der einen Matrix mit der *zweiten Spalte* der anderen Matrix elementweise multipliziert und summiert, um das zweite Element in der ersten Zeile von

[26] Ein verbreitetes Format ist Compressed Sparse Row (CSR), bei dem die vorhandenen Werte mit ihren Indizes gespeichert werden, für einen Überblick über verschiedene Kompressionsverfahren siehe Wikipedia (2022b; https://en.wikipedia.org/wiki/Sparse_matrix).

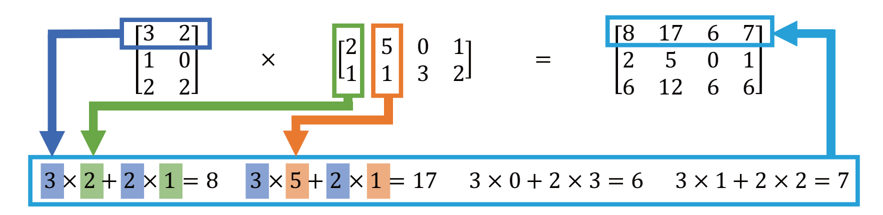


Abb. 4.11 Rechenweg bei der Matrixmultiplikation. Die Farben und Pfeile illustrieren, aus welcher Zeile bzw. Spalte einzelne Werte in der Berechnung stammen. (Quelle: eigene Darstellung)

Matrix C zu ermitteln. Dieses Vorgehen wird für alle weiteren Zeilen und Spalten von Matrix A und B wiederholt (Abb. 4.11). Anders als bei der einfachen Multiplikation von zwei Zahlen, kann man die Operanden A und B nicht einfach vertauschen (= nicht kommutativ) – selbst wenn die Bedingung erfüllt sein sollte, dass die Spaltenanzahl von A der Zeilenanzahl von B entspricht, ergibt das ein anderes Ergebnis. Probieren Sie es einmal aus!

Um nun die Ähnlichkeit zwischen zwei Texten, Personen oder anderen Vektoren zu berechnen, muss man noch einen Schritt tiefer in die Matrizenmultiplikation eintauchen und eine weitere Operation heranziehen: das Transponieren von Matrizen. Hierbei wird eine Matrix A so gedreht, dass die Zeilen zu Spalten und die Spalten zu Zeilen werden. Die transponierte Matrix wird als A' bezeichnet. Da Spalten und Zeilen dann automatisch der oben angesprochenen Bedingung entsprechen, kann eine Matrix immer auch mit der transponierten Matrix multipliziert werden. Standardisiert man die Matrizen vorher, erhält man ein weitverbreitetes und vielseitig einsetzbares Ähnlichkeitsmaß, die Kosinusähnlichkeit (Abb. 4.12):[27]

1. Nimmt man den oben in Abb. 4.9 dargestellten Datensatz, berechnet für jede Zeile die Summe der quadrierten Werte und zieht daraus die Wurzel, entspricht das der sogenannten L2-Norm. Die Zahl drückt die Gesamtzahl der Reaktionen auf einen Post aus, wobei umfangreiche Reaktionen besonders stark gewichtet sind.
2. Mittels Division aller Werte durch die L2-Norm, werden die Zellenwerte auf den Bereich 0 bis 1 normalisiert. Das Resultat ist ein Wert, der ausdrückt wie charak-

[27] Das Beispiel wird hier gewählt, um es leicht nachvollziehbar zu machen. Genau genommen müsste zunächst geklärt werden, inwiefern die Anzahl der Likes, Kommentare und Shares überhaupt vergleichbare Metriken sind.

1. Wurzel der Quadratsumme (L2)

	likes	comments	shares	L2-Norm
Post 1	0	6	2	6,3
Post 2	20	0	8	21,5
Post 3	6	0	0	6,0
Post 4	5	0	0	5,0
Post 5	0	4	1	4,1
Post 6	64	0	1	64,0

2. Normalisieren

	likes	comments	shares
Post 1	0,0	0,9	0,3
Post 2	0,9	0,0	0,4
Post 3	1,0	0,0	0,0
Post 4	1,0	0,0	0,0
Post 5	0,0	1,0	0,2
Post 6	1,0	0,0	0,0

3. Transponieren

Post 1	Post 2	Post 3	Post 4	Post 5	Post 6
0,0	0,9	1,0	1,0	0,0	1,0
0,9	0,0	0,0	0,0	1,0	0,0
0,3	0,4	0,0	0,0	0,2	0,0

4. Multiplizieren

	Post 1	Post 2	Post 3	Post 4	Post 5	Post 6
Post 1	1,00	0,12	0,00	0,00	1,00	0,00
Post 2	0,12	1,00	0,93	0,93	0,09	0,93
Post 3	0,00	0,93	1,00	1,00	0,00	1,00
Post 4	0,00	0,93	1,00	1,00	0,00	1,00
Post 5	1,00	0,09	0,00	0,00	1,00	0,00
Post 6	0,00	0,93	1,00	1,00	0,00	1,00

Abb. 4.12 Berechnung der Kosinusähnlichkeit. (Quelle: eigene Darstellung)

teristisch eine bestimmte Reaktion für den Post ist. Zum Beispiel bestehen alle Reaktionen (1,0) von Post 6 aus Likes, dieser Post erhält keine Kommentare (0,0).
3. Transponiert man die normalisierte Matrix, so werden die Posts zu Spalten.
4. Multipliziert man die normalisierte Matrix mit der transponierten normalisierten Matrix, ergibt sich bei Betrachtung der höchsten Werte ein Muster.

Das Ergebnis ist eine Matrix mit Posts in den Zeilen und den gleichen Posts in den Spalten. Die Werte an den Kreuzungen stellen ein Ähnlichkeitsmaß dar: Je höher der Wert, umso ähnlicher sind sich die Aktivitätsschemata dieser Posts. Der erste und fünfte Post gleichen sich darin, dass sie keine Likes, dafür aber andere Reaktionen erhalten haben. Andere Posts sind sich dahingehend ähnlich, dass die Likes augenscheinlich eine stärkere Rolle im Aktivitätsmuster spielen.

Eine solche Aufbereitung von Daten führt Schritt für Schritt zum Destillieren von Mustern. Und wenn Sie tatsächlich etwas nachgerechnet haben, wird auch klar: Per Hand ist das aufwendig. In den nächsten Kapiteln lernen Sie, wie Sie Computer für sich arbeiten lassen.

Übungsfragen

1. Was unterscheidet das Long-Format vom Wide-Format?
2. Was unterscheidet einen Left Join vom Inner Join und vom Full Join?
3. Wofür kann das Map-Reduce-Verfahren eingesetzt werden?
4. Wie werden die einzelnen Werte in einer Matrix adressiert?
5. Nennen Sie einen Anwendungsfall für die Matrixmultiplikation!

Weiterführende Literatur

Beaulieu, A. (2021). *Einführung in SQL. Daten erzeugen, bearbeiten und abfragen.* (3. Aufl.). Heidelberg: O'Reilly.

Becher, M. (2011). *XML. DTD, XML-Schema, XPath, XQuery, XSLT, XSL-FO, SAX, DOM.* Herdecke, Dortmund: W3L-Verlag.

Cleve, J. & Lämmel, U. (2014): *Data Mining*. München: Oldenbourg.

Fitzgerald, M. (2012). *Einstieg in reguläre Ausdrücke. Schritt für Schritt reguläre Ausdrücke verstehen.* Beijing: O'Reilly.

Freeman, A. (2011). *The definitive guide to HTML5. All you need to know to use HTML5 professionally.* New York: Apress Springer.

Friedl, J. E. F. (2006). *Mastering regular expressions.* (3. Aufl.). Sebastapol: O'Reilly.

Kirchgessner, K. (2012). *Vektor- und Matrizenrechnung für Dummies.* Weinheim: John Wiley & Sons Incorporated.

Meyer, E. A. (2019). *CSS. kurz & gut.* (5. Aufl.). Heidelberg: dpunkt.verlag.

Schicker, E. (2017). *Datenbanken und SQL. Eine praxisorientierte Einführung mit Anwendungen in Oracle, SQL Server und MySQL.* (5. Aufl.). Wiesbaden: Springer Vieweg.

Stubblebine, T. (2004). *Reguläre Ausdrücke. kurz & gut.* Beijing: O'Reilly.

Wickham, H. & Grolemund, G. (2016). *R for Data Science.* Sebastopol: O'Reilly UK Ltd.

Literatur

Apache Software Foundation. (2022a). Apache Hadoop (Version 3.3.3) [Computer software]. https://hadoop.apache.org/

Chomsky, N. (1956). Three models for the description of language. *IEEE Transactions on Information Theory*, *2*(3), 113–124. https://doi.org/10.1109/TIT.1956.1056813

DB Browser for SQLite. (2022). DB Browser for SQLite. The Official home of the DB Browser for SQLite [Computer software]. https://sqlitebrowser.org/

Fayyad, U., Piatetsky-Shapiro, G. & Smyth, P. (1996). From Data Mining to Knowledge Discovery in Databases. *AI Magazine*, *17*(3), 37–54. https://doi.org/10.1609/aimag.v17i3.1230

Feasel, J. (2021). *SQL Fiddle*. Zugriff am 30.05.2022. http://sqlfiddle.com/

Golem. (2022). *IT-News für Profis*. Zugriff am 01.06.2022. https://www.golem.de/

Google. (2022c). Google Tabellen [Computer software]. https://spreadsheets.google.com

Goyvaerts, J. (2021). *Regular-Expressions.info. The Premier website about Regular Expressions*. Zugriff am 30.05.2022. https://regular-expressions.info/

Imfernsehen. (2022). *Serien A-Z. R*. Zugriff am 30.05.2022. https://www.fernsehserien.de/serien-a-z/r

Jünger, J. & Keyling, T. (2022). Facepager. An application for automated data retrieval on the web. (Version 4.4.4) [Computer software]. https://github.com/strohne/Facepager/

Kleene, S. C. (1956). Representation of Events in Nerve Nets and Finite Automata. In C. E. Shannon & J. McCarthy (Hrsg.), *Automata Studies.* (S. 3–42). Princeton: Princeton University Press. https://doi.org/10.1515/9781400882618-002

Kuchling, A. M. (Python Software Foundation, Hrsg.). (2017). *Regular Expression HOWTO. The Backslash Plague*. Zugriff am 30.05.2022. https://docs.python.org/3.3/howto/regex.html#the-backslash-plague

MariaDB Foundation. (2022). MariaDB Server. The open source relational database (Version 10.9.0) [Computer software]. https://mariadb.org/

Oracle. (2021). MySQL (Version 8.0) [Computer software]. https://www.mysql.com/

Oracle. (2022b). *MySQL Documentation*. Zugriff am 30.05.2022. https://dev.mysql.com/doc/

Python Software Foundation. (2022a). *Lexical analysis. String and Bytes literals*. Zugriff am 30.05.2022. https://docs.python.org/3/reference/lexical_analysis.html#string-and-bytes-literals

Ricci, F., Rokach, L. & Shapira, B. (Hrsg.). (2015). *Recommender Systems Handbook*. Boston: Springer. https://doi.org/10.1007/978-1-4899-7637-6

Richardson, L. (2022). Beautiful Soup [Computer software]. https://www.crummy.com/software/BeautifulSoup

Robinson, D., Bryan, J. & Elias, J. (2020). fuzzyjoin: Join Tables Together on Inexact Matching (Version 0.1.6) [Computer software]. https://cran.r-project.org/package=fuzzyjoin

Selfhtml. (2021). *XML/XSL/XPath*. Zugriff am 30.05.2022. https://wiki.selfhtml.org/wiki/XML/XSL/XPath

Selfhtml. (2022). *CSS/Tutorials/Selektoren*. Zugriff am 30.05.2022. https://wiki.selfhtml.org/wiki/CSS/Tutorials/Selektoren

Silva, V. (2008). *What is the best regular expression to check if a string is a valid URL?*, Stack Overflow. Zugriff am 30.05.2022. https://stackoverflow.com/questions/161738/what-is-the-best-regular-expression-to-check-if-a-string-is-a-valid-url

Skinner, G. (2021). *RegExr. RegExr is an online tool to learn, build, & test Regular Expressions*. Zugriff am 30.05.2022. https://regexr.com/

SQLite Development Team. (2022). SQLite [Computer software]. https://www.sqlite.org/

Taylor & Francis. (2022). *Media, Cultural & Communication Studies Title List 2019*. https://librarianresources.taylorandfrancis.com/collection/media-cultural-communication-studies-collection/media-cult-com-stu/

Thompson, K. (1968). Programming Techniques: Regular expression search algorithm. *Communications of the ACM*, *11*(6), 419–422. https://doi.org/10.1145/363347.363387

W3Schools. (2022d). *SQL Tutorial*. Zugriff am 30.05.2022. https://www.w3schools.com/sql/

Wickham, H. (2014). Tidy Data. *Journal of Statistical Software*, *59*(10). https://doi.org/10.18637/jss.v059.i10

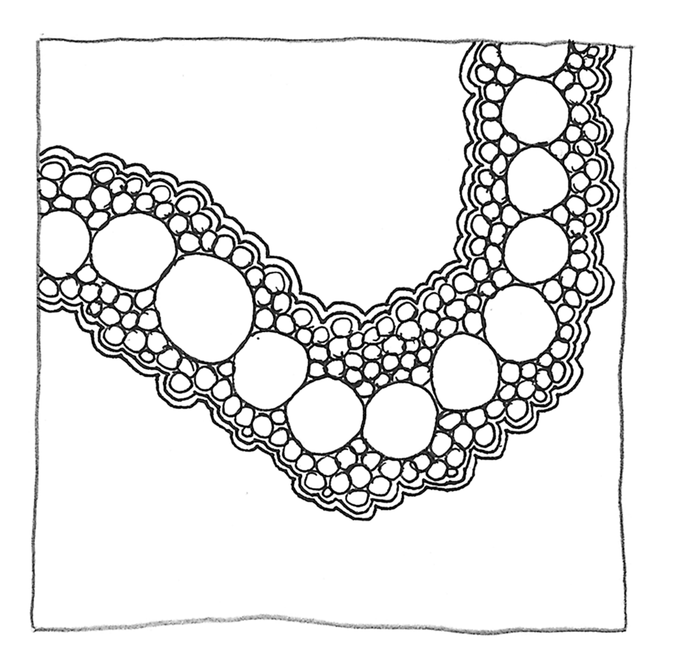
Wikipedia. (2022b). *Sparse matrix*. Zugriff am 30.05.2022. https://en.wikipedia.org/wiki/Sparse_matrix

Teil II

Programmierkonzepte



5 Programmierung



J. Jünger, C. Gärtner, *Computational Methods für die Sozial- und Geisteswissenschaften*, https://doi.org/10.1007/978-3-658-37747-2_5

Zusammenfassung

Dieses Kapitel führt in die Programmierung mit R und Python ein. Sie erlernen die Grundlagen der beiden Programmiersprachen, um erste Erfahrungen mit der Datenaufbereitung und -analyse zu sammeln.

Im Online-Repositorium unter https://github.com/strohne/cm finden Sie begleitend zum Kapitel weitere Materialien, auf die wir im Text mit ☛ verweisen.

Schlüsselwörter

R · RStudio · Tidyverse · ggplot · Python · Jupyter Notebook · Pandas · Numpy

Computer sind aus wissenschaftlichen Projekten kaum wegzudenken. Sie helfen bei der Verwaltung von Daten, nehmen klaglos Routineaufgaben der Datenanalyse ab und sind meist auch an der Kommunikation von Ergebnissen beteiligt. Ohne die Anweisungen von Menschen, die in Form von Programmen gegeben werden, sind Computer aber nutzlos. Bei der Benutzung aktueller PCs oder Smartphones treten die Programme in der Regel in den Hintergrund, sie verbergen sich hinter der Benutzeroberfläche. In diesem Kapitel geht es darum, hinter die Oberfläche zu schauen und selbst Programme zu schreiben. Das ist besonders dann sinnvoll, wenn es für ein bestimmtes Problem entweder keine vorkonfektionierte Software gibt oder wenn die Prozesse der Datenerhebung und -analyse möglichst transparent und kontrolliert sein sollen. Denn die Programme dokumentieren im besten Fall jeden einzelnen Schritt der eigenen Vorgehensweise. Das gilt insbesondere für sogenannte Skripte, die eine Abfolge von Befehlen in einem Textdokument festhalten. Diese Skripte sind von Menschen und von Maschinen lesbar. Auf einem Computer wird dazu ein Interpreter – das ist ebenfalls ein Programm – eingesetzt, der die Befehle Zeile für Zeile abarbeitet. Programme führen also Programme aus.

Eine solche Interpretersprache für statistische Berechnungen ist R. Demgegenüber stehen Compilersprachen wie C, bei denen der Quelltext vor der Ausführung des Programms erst in Maschinensprache übersetzt wird. Die Kompilierung des Programms braucht dann einen Moment, dafür laufen die fertigen Programme aber schneller ab. Zwischen Interpreter- und Compilersprachen stehen Sprachen wie Python oder Java. Hier wird erst ein sogenannter Bytecode erzeugt, der dann von einem Interpreter abgearbeitet wird.

Programmiersprachen lassen sich also danach einteilen, wie aus dem Quelltext ein lauffähiges Programm entsteht. Sie unterscheiden sich aber auch in ihrer Art und Weise der Problemlösung. Wird einfach nur ein Befehl nach dem anderen aufgeschrieben und abgearbeitet, so handelt es sich um prozedurale bzw. imperative Programmierung. Für komplexere Abläufe werden die Befehle in Kontrollstrukturen eingebettet, sodass etwa Anweisungen in einer for-Schleife wie „für alle Zahlen kleiner als 5“ nacheinander abgearbeitet werden. Dagegen wird das Problem bei funktionaler Programmierung so formuliert, dass es auf eine Formel gebracht wird. Hier sind in einer Codezeile häufig mehrere Funktionsaufrufe ineinander verschachtelt. Im Vergleich zur imperativen Programmierung können Befehle demnach kürzer und präziser gehalten werden, sind aber dadurch auch abstrakter. Bei objektorientierter Programmierung wiederum werden die Probleme als Klassen abgebildet, die Eigenschaften und Methoden besitzen. Von einer solchen Klasse – zum Beispiel einem Balkendiagramm – lassen sich dann Instanzen erzeugen, also etwa ein konkretes Balkendiagramm mit blauen und roten Balken.[1]

Aktuelle Programmiersprachen verbinden in der Regel mehrere dieser Konzepte. Das gilt auch für die im Folgenden besprochenen Sprachen R und Python. Prozedurale Elemente sind in fast jedem Skript vorhanden. Besonders in R sind immer wieder auch funktionale Elemente offensichtlich und in Python kommt man häufig mit Klassen und Objekten in Berührung. Jede dieser Programmiersprachen bringt eine eigene Perspektive mit sich – die sich als Meme zusammenfassen lassen und so zur Reflexion über sinnvolle Einsatzszenarien einladen (Abb. 5.1). Auch wenn sich mit R prinzipiell beliebige Probleme lösen lassen, ist es vor allem für die Datenanalyse entworfen und darauf zugeschnitten. Zu Beginn erscheint die Sprache jedoch etwas unübersichtlich. Python verfügt über eine eingängige Syntax und wird sehr universell eingesetzt, nicht nur für die Datenanalyse. Die Einarbeitung in diese Programmiersprachen lohnt sich vor allem deshalb, weil man damit die Grenzen spezialisierter Tools wie Excel oder SPSS überwinden kann (Dahley 2017; Guy_Jantic 2019).

[1] Zu den Vorzügen objektorientierter Programmierung bei der Erstellung von Grafiken siehe Wilkinson (1999, S. 2–19).

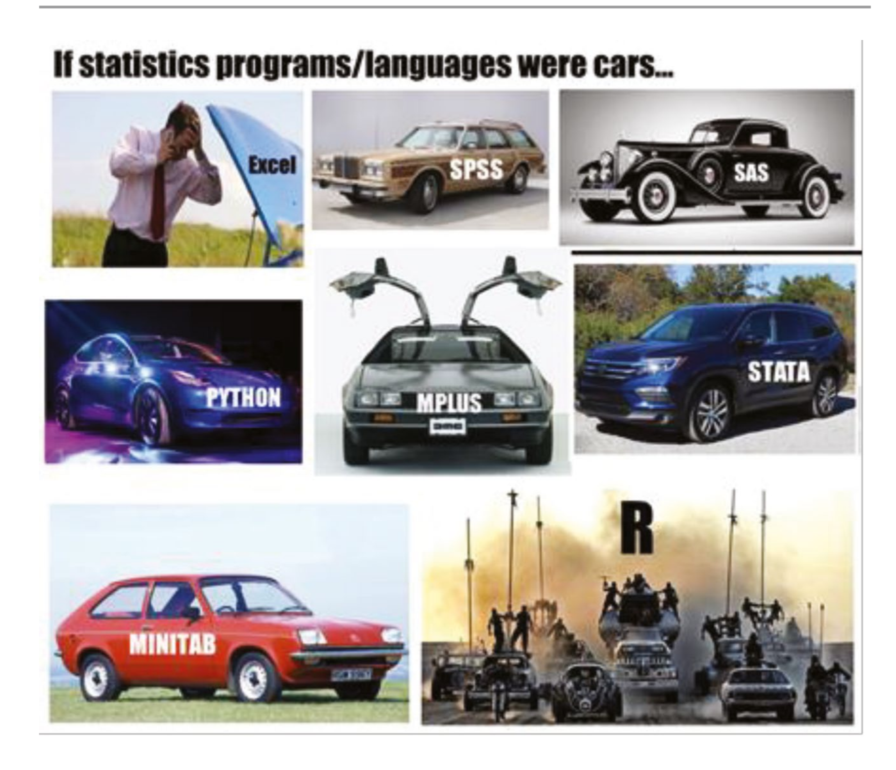


Abb. 5.1 Wenn statistische Programme und Sprachen Autos wären. (Quelle: Dahley 2017; Guy_Jantic 2019)

5.1 Einführung in die Datenanalyse mit R

R ist eine Programmiersprache, die für die Datenanalyse optimiert ist. Im Gegensatz zu vielen kommerziellen Produkten wie Stata oder SPSS können Sie R kostenlos verwenden – und Sie profitieren darüber hinaus von einer aktiven Entwicklungsgemeinschaft, in die Sie sich nach einiger Zeit sogar selbst einbringen können.

Während sich klassische Statistikprogramme durch eine Benutzeroberfläche auszeichnen und man erst bei speziellen Problemstellungen auf die Programmierung (in der Regel über eine sogenannte Syntax) ausweicht, dreht sich dieses Verständnis bei R um. Denn R ist eine Programmiersprache ohne Benutzeroberfläche,

die primär über eine Kommandozeile bedient wird. Grafische Programme binden dann R ein, um die Arbeit zu erleichtern. Das klingt vielleicht im ersten Moment kompliziert, diese Arbeitsweise hat aber drei ganz wesentliche Vorzüge. Sie können den Stand eines Projekts inklusive der Daten und der Schritte für die Datenanalyse jederzeit abspeichern, weitergeben und wiederholen (replizieren). Zudem können Sie klein anfangen und das Projekt nach und nach erweitern, ohne immer wieder von vorne zu beginnen (inkrementell). Und schließlich können Sie bei Bedarf Teile der Datenauswertung automatisieren, um zum Beispiel die gleichen Schritte mit wenig Aufwand auf weitere Datensätze anzuwenden (skalieren).

Tatsächlich wird die Arbeit mit R (R Core Team 2022) erheblich erleichtert, wenn man dafür eine Entwicklungsumgebung verwendet. Eine für die Datenanalyse häufig eingesetzte Entwicklungsumgebung ist RStudio (RStudio 2022a). Die kostenlose Version von RStudio steht unter einer Open-Source-Lizenz (AGPL) und bringt mehr Funktionen mit als für die meisten Einsatzzwecke im sozial- und geisteswissenschaftlichen Kontext benötigt werden.

Installieren Sie beides nacheinander auf Ihrem Computer:

- Die base-Distribution von R: https://www.r-project.org/ (Startseite) bzw. https://cran.r-project.org/ (Downloadseite)
- RStudio Desktop: https://posit.co/download/rstudio-desktop/

5.1.1 Die Oberfläche von RStudio

Mit RStudio können Sie die verschiedenen Dateien eines Projekts in einem gemeinsamen Ordner auf der Festplatte verwalten. Legen Sie dazu über das FILES-Menü ein neues leeres Projekt in einem Ordner Ihrer Wahl an. Sobald Sie dieses Projekt angelegt haben, beziehen sich alle ausgeführten Befehle, beispielsweise zum Dateneinlesen oder -abspeichern, auf dieses Arbeitsverzeichnis. Gewöhnen Sie sich von vornherein an, immer in einem Projekt zu arbeiten – ganz oben rechts in RStudio sollten Sie den Namen des aktuell geöffneten Projektes sehen.

Der wichtigste Dateityp innerhalb eines Projekts sind R-Skripte, in denen die Programmierbefehle gespeichert werden. Legen Sie innerhalb des Projekts ein neues Skript an, Sie finden den Befehl dazu ebenfalls im FILE-Menü (Abb. 5.2).

Anschließend sollten Sie eine Oberfläche mit vier Bereichen sehen, die bei Bedarf umsortiert werden kann (Abb. 5.3).

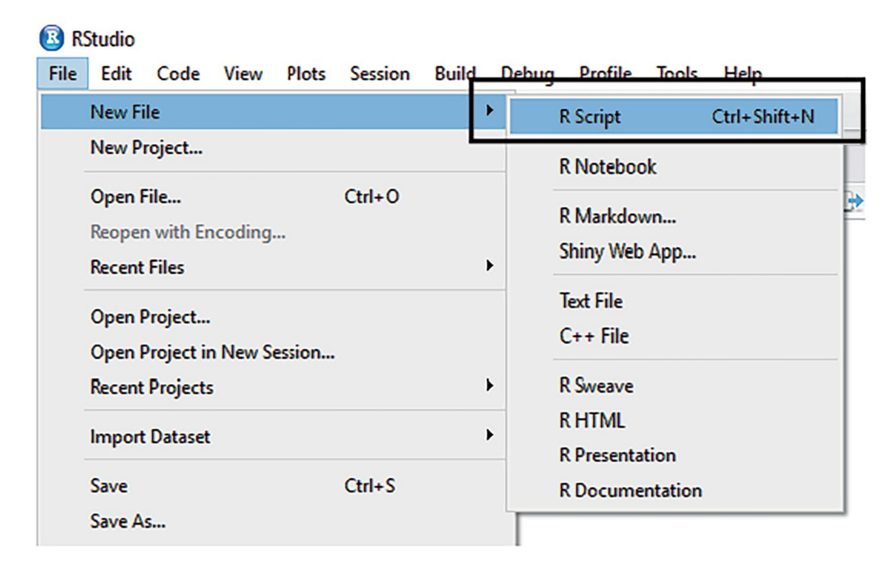


Abb. 5.2 Erstellen eines neuen R-Skripts. (Quelle: eigene Darstellung)

Skripte und Datensätze

Aktuelle Datenobjekte

Konsole

Dateien, Hilfe und Grafiken

Abb. 5.3 Layout von RStudio. (Quelle: eigene Darstellung)

Links oben werden Skripte angezeigt. Ein Skript enthält alle Befehle für die Datenanalyse. Mit der Schaltfläche SOURCE wird die gesamte Datei ausgeführt – das ist aber meistens erst am Ende der Entwicklung sinnvoll. Vorerst geben Sie hier einzelne Codezeilen ein, markieren diese mit der Maus oder Tastatur und führen nur den markierten Bereich über die Schaltfläche RUN oder schneller noch mit der Tastenkombination Strg + Enter (Windows) bzw. Command + Enter (Mac) aus. Wenn Sie nichts markieren, wird die aktuelle Zeile ausgeführt, in der sich der Cursor befindet. Der Cursor springt anschließend auf die nächste Zeile, sodass Sie sich Stück für Stück durch den geschriebenen Code arbeiten können.

Beim Ausführen wird der Programmcode an die Konsole im unteren Bereich übergeben. Probieren Sie es aus, indem Sie die folgenden Zeilen in das neue Skript schreiben, den Cursor auf die erste Zeile setzen, die Strg- bzw. Command-Taste gedrückt halten und dreimal nacheinander die Enter-Taste betätigen:

```
ich <- "Leo"
gruss <- paste0("Hallo", ich)
print(gruss)
```

Vielleicht ahnen Sie bereits, was diese Zeilen bewirken, eine Erläuterung folgt unten.

Links unten befindet sich die Konsole (engl. *console*), so wird die R-Kommandozeile genannt, in welcher der Inhalt des Skripts ausgeführt wird. Die Konsole zeigt sowohl Fehler als auch zurückgegebene Werte an. Sie können Befehle auch direkt in die Konsole statt in das Skript eingeben. Wenn Sie den Namen eines Objekts eingeben, wird wie mit dem `print()` -Befehl der Inhalt des Objekts angezeigt:

```
gruss
```

Der Vorteil von Skripten ist, dass Sie Befehle schrittweise aneinanderreihen, umsortieren und weiterentwickeln können. In der Konsole müssten Sie jeden Befehl, um ihn erneut auszuführen, auch erneut eingeben.

Rechts oben wird der Arbeitsbereich (engl. *environment*) angezeigt. Dieser umfasst alle Objekte und Funktionen, die Sie mit einem Skript geladen oder erzeugt haben. Nach dem Ausführen der obigen Zeilen sehen Sie im Arbeitsbereich die zwei Objekte `gruss` und `ich`. Diese Objekte haben Sie mit dem Zuweisungsoperator <− erzeugt. Hiermit wird dem Symbol vor dem Operator der Wert dahinter zugewiesen. Mit Symbol ist der Name eines Objekts gemeint, den Sie selbst fest-

legen können.[2] Sie können alle Objekte im Arbeitsbereich mit der Besen-Schaltfläche löschen. Einzelne Objekte werden über den `rm()`-Befehl im Skript oder in der Konsole gelöscht. Probieren Sie es aus:

```
rm(ich)
```

Über der Liste finden Sie eine Schaltfläche, um Datensätze (CSV-, Exceldateien etc.) einzulesen. Später werden Sie Programmierbefehle kennenlernen, mit denen Datensätze eingelesen werden – das ist in der Regel der bessere Weg, weil die Skripte dann auch von anderen oder einige Zeit später besser nachvollzogen werden können, da die verwendeten Datensätze aus dem Skript ersichtlich sind. Sobald Datensätze aufgelistet sind, können Sie diese anklicken und sie werden in einem neuen Bereich links oben neben Ihren geöffneten Skripten angezeigt.

Rechts unten finden Sie unter anderem die FILES-Ansicht. Dort sehen Sie die Dateien des Projekts. Es handelt sich hierbei um das Arbeitsverzeichnis auf Ihrem Computer, in dem auch das R-Projekt liegt. In der PLOTS-Ansicht werden Grafiken ausgegeben. Im PACKAGES-Bereich können Sie zusätzliche Programmbibliotheken verwalten und vor allem installieren. Packages sind Sammlungen von Funktionen, die von anderen Personen aus der R-Entwicklungscommunity zusammengestellt wurden. Damit Sie die Funktionen aus einem Package verwenden können, müssen Sie das Package zunächst unter INSTALL heraussuchen und installieren. Die installierten Packages sind in der User Library mit ihrer Versionsnummer aufgelistet. Da die Funktionen in Packages fortlaufend weiterentwickelt werden, kann es sein, dass die Versionen mit der Zeit veralten. In diesem Fall können Sie die Packages mit UPDATE auf den neuesten Stand bringen.

Zwei Packages, die für viele Projekte sehr hilfreich sind und weiter unten besprochen werden, sind *tidyverse* und *skimr*. Sie können beide gleich installieren, damit sie für die anderen Beispiele in diesem Buch bereitstehen. Nachdem Sie ein Package installiert haben, müssen Sie es über den Befehl `library()` aktivieren:

```
library(tidyverse)
```

In der Liste sind geladene Packages an einem Häkchen erkennbar. Packages müssen bei jedem Öffnen von R zwar nicht immer wieder neu installiert, aber vor

[2] Sie könnten alternativ auch das =-Zeichen verwenden, in R ist dafür aber der nach links gerichtete Pfeil üblich und in einigen Fällen sogar notwendig.

der Arbeit mit dem `library()`-Befehl erneut geladen werden. Erst dadurch weiß R, in welchen Packages die im Skript verwendeten Funktionen zu finden sind.

Der VIEWER zeigt den Inhalt von Textdateien oder HTML-Dateien an. Ganz besonders wichtig ist aber der Hilfebereich. Die eingebaute Hilfe ist eine große Stärke von R. Sie können zu jedem Befehl die Dokumentation aufrufen, indem Sie dem Befehl im Skript ein Fragezeichen voranstellen und die entsprechende Zeile ausführen. In der Hilfe werden die möglichen Argumente beschrieben und auch Beispiele gegeben. Probieren Sie es für den Befehl `paste0()` aus:

```
?paste0
```

Noch schneller geht das, wenn Sie den Cursor auf dem Befehl platzieren und die F1-Taste drücken. Die Hilfe wird nur für die Befehle angezeigt, die in aktuell geladenen Packages enthalten sind. Sie können aber auch die gesamte Hilfe durchsuchen, inklusive zwar installierter, aber noch nicht geladener Packages, indem Sie zwei Fragezeichen verwenden. So können Sie zum Beispiel herausfinden, in welchem Package ein Befehl enthalten ist – zum Beispiel finden Sie so das *skimr*-Package, wenn es zwar installiert, aber nicht geladen ist:

```
??skim
```

Um zu erkunden, was alles in einem Package enthalten ist, klicken Sie in RStudio in der Liste aller Packages auf das jeweilige Package. Sie können sich dann zunächst einen Überblick über die Funktionen verschaffen. Viele Packages stellen zudem Einführungen zur Verfügung, die sogenannten Vignetten. Klicken Sie dazu in der Dokumentation auf den Punkt „User guides, package vignettes and other documentation“. Zu Beginn werden Ihnen die Ausführungen in der Hilfe zwar kryptisch vorkommen, es lohnt sich dennoch, sich schrittweise damit vertraut zu machen. Lassen Sie sich nicht abschrecken – die Hilfe wird nach und nach zu einem Ihrer wichtigsten Hilfsmittel werden.

5.1.2 Der Umgang mit Skripten

Aufbau eines Skripts Ein R-Skript (☛ *Repositorium*) besteht aus einer Abfolge von Zuweisungen, Funktionsaufrufen und Kommentaren.[3] Mit Zuweisungen werden Daten sozusagen in verschiedene Schachteln, Dosen und Tüten wegsortiert. Funktionen

[3] Siehe auch die R Language Definition (R Core Team 2021).

verarbeiten die Daten, formen sie um, zerlegen oder kombinieren sie. Kommentare sind Notizzettel, die man an Daten und Funktionen anheftet. So kann man sich notieren, was in welcher Schachtel enthalten ist und wozu eine Funktion dient.

Genauer gesagt werden bei Zuweisungen Werte in einem Objekt gespeichert, das über ein Symbol bezeichnet wird. Wenn im Folgenden von Objekten, Symbolen oder Variablen gesprochen wird, können diese Begriffe meist ausgetauscht werden, auch wenn sie nicht vollständig synonym sind. Für eine Zuweisung wird der linksgerichtete Pfeil `<-` eingesetzt. Dieses Zuordnungszeichen kann in RStudio über die Tastenkombination Alt + Minustaste (Windows) bzw. Optionstaste + Minustaste (Mac) erzeugt werden, es kann aber auch mit dem Kleinerzeichen und einem Minuszeichen eingetippt werden.

```
name <- "Bea"
```

Zur Verarbeitung von Daten werden Funktionen eingesetzt, die in Klammern Argumente entgegennehmen. Diese Argumente sind der Input der Funktion. In der Funktion werden die Daten verwendet und ein Output wird zurückgeben. Dieser Output kann einem Objekt zugewiesen werden. Die `paste0()`-Funktion nimmt beispielsweise mehrere Zeichenketten als Input entgegen und verbindet sie zu einer einzigen Zeichenkette, die dann als Ergebnis in einem neuen Objekt abgespeichert werden kann:

```
name <- paste0("Inger", " Engmann")
```

Ähnlich funktioniert die Datenverarbeitung mithilfe von Operatoren, etwa mit den Grundrechenarten. Im Unterschied zum vorherigen Funktionsaufruf wird der Funktionsname nicht vor (Präfixnotation), sondern zwischen die Argumente gesetzt (Infixnotation):

```
alter <- 2017 - 1989
jahr <- 1989 + alter
```

Beliebige Funktionen können mit dem folgenden Schema selbst definiert werden (führen Sie zur Definition der Funktion den gesamten Funktionsblock aus):

```
calculate_age <- function(year_now, year_birth) {
  alter <- year_now - year_birth
  return (alter)
}
```

Der Name der Funktion steht links vom Zuweisungszeichen – Sie können sich selbst einen Namen bestehend aus Buchstaben und Ziffern ausdenken. Häufig kommen auch der Punkt oder der Unterstrich in Funktionsnamen vor. Die Parameter werden in Klammern angegeben. Innerhalb der geschweiften Klammern folgt der Code, welcher in der Funktion ausgeführt werden soll. Am Ende wird mit `return()` ein Wert zurückgegeben.[4] Diese Funktion kann nun über den Namen verwendet werden:

```
alter <- calculate_age(2017, 1989)
```

Selbst Funktionen zu schreiben ist besonders sinnvoll, wenn es nicht bereits eine Funktion für den gewünschten Zweck gibt, oder wenn man die gleichen, aufeinanderfolgende Befehle an mehreren Stellen im Skript benötigt. Damit man sich nicht stetig wiederholt[5] und das Skript möglichst übersichtlich bleibt, wird der Code in eine Funktion gekapselt. Das erhöht die Lesbarkeit für andere Forscher:innen. Und auch wenn zu einem späteren Zeitpunkt etwas an den Funktionen verändert werden soll, muss nicht jede entsprechende Zeile im Skript, sondern nur die Funktion angepasst werden. Für viele Zwecke gibt es Funktionen, die andere Entwickler:innen bereits geschrieben haben und über Packages bereitstellen. Sobald Sie ein Package mit `library()` geladen haben (siehe oben), können Sie die darin enthaltenen Funktionen aufrufen und verwenden. Wenn Sie gerade mit dem Erlernen einer Programmiersprache beginnen, werden Sie die Funktionen zunächst recherchieren müssen, beispielsweise über eine Suchmaschine oder die Hilfe. Mit der Zeit werden Sie mehr und mehr Funktionen auswendig kennen, sodass das Programmieren dann leichter von der Hand geht.

Kommentare sind ein weiteres wichtiges Element in Skripten. Sie beginnen mit einem Rautezeichen und helfen dabei, den Code zu beschreiben, damit man ihn später besser versteht:

```
# Dies ist ein Kommentar
```

In RStudio lässt sich ein Skript mit Kommentaren in einzelne Abschnitte einteilen, indem an eine Kommentarzeile mindestens vier Bindestriche angehängt werden (Abb. 5.4). Die so gekennzeichneten Zeilen werden in ein Inhaltsverzeichnis aufgenommen. Dieses Inhaltsverzeichnis lässt sich über die Outline-Buttons rechts oben oder unterhalb des Skripts einblenden, sodass einzelne Abschnitte schnell angesprungen werden können.

---

[4] Ohne das `return`-Statement wird automatisch das letzte Objekt zurückgegeben.

[5] Dieses Prinzip wird als dry (don't repeat yourself) bezeichnet.

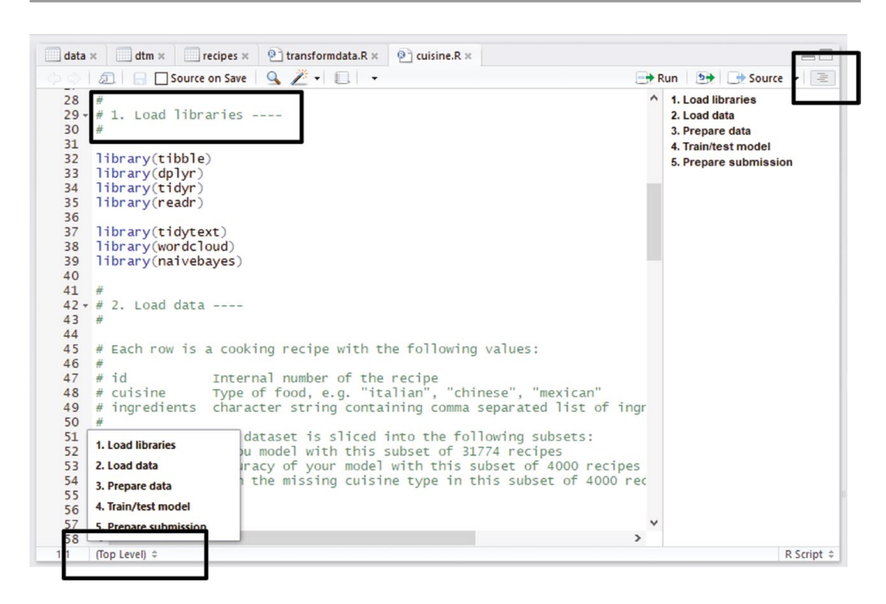


Abb. 5.4 Gliederung eines R-Skripts in Abschnitte. (Quelle: eigene Darstellung)

Die Dokumentation von Code ist wichtig, denn kaum jemand kann sich noch Monate später genau erinnern, warum welcher Befehl wo eingefügt wurde. Zusätzlich ist wie beim Verfassen von Romanen oder Aufsätzen auch beim Programmieren ein konsistenter Schreibstil (engl. *coding style*) für die Verständlichkeit sehr hilfreich. Sie sollten sich beispielsweise entscheiden, ob Sie einzelne Teile von Bezeichnern mit Punkten, Unterstrichen oder CamelCase trennen wollen. Alle diese Varianten und noch einige mehr sind in R möglich. Jedoch können sie auch schnell zu Verwirrung oder Fehlern führen, denn bei unterschiedlicher Schreibweise sind es auch unterschiedliche Bezeichner:

```
calculate.age
calculate_age
calculateAge
CalculateAge
```

Sie können sich zum Beispiel am Tidyverse Style Guide orientieren.[6] Dort wird für Variablen und Funktionen die Schreibweise mit Unterstrichen empfohlen.

[6] Siehe Wickham (2021; https://style.tidyverse.org/).

Die Ausgabe in der Konsole Beim Programmieren macht man Fehler, das ist nahezu unvermeidlich. Dazu kommt, dass jeder einzelne Computer seine Eigenarten hat, die schnell zu Stolpersteinen werden. Deshalb besteht ein wesentlicher Teil der Entwicklung in der Fehlersuche und -behebung. Das ist im ersten Moment ärgerlich – es führt aber gleichzeitig immer wieder zu kleinen Erfolgserlebnissen, wenn ein Fehler behoben ist.

Ein Programmierfehler wird auch Bug genannt und die Fehlersuche Debugging (siehe Abschn. 6.2). Programmierumgebungen wie RStudio stellen dafür verschiedene Werkzeuge bereit. Insbesondere kann man sich Schritt für Schritt an die fehlerhafte Stelle herantasten und nicht nur tief in den Quellcode der eigenen Skripte eintauchen, sondern auch die Funktionen in Paketen erkunden. Die Arbeit mit diesen Werkzeugen ist zum Beispiel in der Dokumentation von RStudio erläutert.[7]

Bevor man richtiges Debugging lernt, ist es sehr hilfreich, sich mit den Ausgaben auf der Konsole näher zu beschäftigen. Hier werden nicht nur die Befehle eingegeben, sondern auch die Ergebnisse aller Prozesse, die R ausführt, sind ersichtlich. Grundsätzlich muss man zwei Sorten von Meldungen unterscheiden:

1. **Warnungen** treten auf, wenn die aktuellen Umstände eines Funktionsaufrufs nicht ganz dem entsprechen, was der Entwickler bzw. die Entwicklerin einer Funktion erwartet hat. Das Programm läuft trotzdem weiter. Sie sollten aber versuchen, die Warnung zu verstehen und abschätzen, welche Folgen sich ergeben. Denn möglicherweise entsprechen die Ergebnisse der Datenanalyse nicht dem, wovon Sie ausgehen. Unkritische Warnungen ergeben sich häufig beim Einbinden von Packages:

```
> library(dplyr)

Attache Paket: 'dplyr'
The following objects are masked from 'package:stats':
    filter, lag
The following objects are masked from 'package:base':
    intersect, setdiff, setequal, union
Warning message:
Paket 'dplyr' wurde unter R Version 3.4.4 erstellt
```

[7] Siehe McPherson (2021; https://support.rstudio.com/hc/en-us/articles/205612627-Debugging-with-RStudio).

Die Warnung in der letzten Zeile ist in der Regel unkritisch und kann meistens ignoriert werden. Sie deutet lediglich darauf hin, dass sich Ihre R-Version von der für das Package verwendeten R-Version unterscheidet. Diese Meldung können Sie gegebenenfalls durch eine Aktualisierung der R-Version beheben.

Praktisch relevant kann aber der Hinweis auf die Maskierung von Objekten sein. Dieser Hinweis entsteht dadurch, dass in verschiedenen Packages Funktionen oder Objekte mit gleichem Namen enthalten sind. Es wird bei mehrdeutigen Namen immer auf das Objekt aus demjenigen Package zurückgegriffen, das zuletzt geladen wurde. Meistens ist das auch gewünscht. Im vorherigen Beispiel würde ein Aufruf von `filter()` die Funktion aus dem gerade geladenen *dplyr*-Package aufrufen. Falls dagegen die Funktion aus dem *stats*-Package aufgerufen werden soll, dann muss der Name des Packages getrennt durch einen *doppelten* Doppelpunkt explizit angegeben werden: `stats::filter()`.

Warnungen können auch bei der Datenanalyse entstehen, zum Beispiel bei der Visualisierung:

```
> ggplot(data, aes(x = day, y = news)) +
   geom_point()

Warning message:
Removed 8 rows containing missing values (geom_point).
```

Bei einer solchen Warnung sollten Sie überlegen, ob sie nachvollziehbar ist und Ihren Erwartungen entspricht. Wenn Ihnen nicht bewusst war, dass in den Daten acht Werte gefehlt haben, dann schauen Sie sich die Daten noch einmal an und gehen Sie der Ursache auf den Grund!

2. **Fehler** führen dazu, dass das Programm abbricht. Hier bleibt Ihnen nichts anderes übrig, als sich auf die Suche nach der Ursache zu machen:

```
> ggplot(data, aes(x = day, y = Fatalities)) +
   geom_point()
Error in FUN(X[[i]], ...) : object 'Fatalities' not found
```

Die Meldung im Beispiel weist darauf hin, dass ein Objekt nicht gefunden wurde. Im besten Fall liegt das einfach an einem Tippfehler, etwa, weil Sie Groß- und Kleinschreibung verwechselt haben. Wenn Sie keine Idee haben, woher der Fehler stammen könnte, dann kopieren Sie die Meldung in die Suchmaschine Ihrer Wahl. Häufig sind Sie nicht der oder die Erste mit einer solchen Meldung und kön-

nen auf Hilfe hoffen. Eine gute Quelle für Hilfestellungen ist die Seite Stack Overflow.[8] Dort können Sie auch selbst Fragen stellen, wenn Sie nicht weiterkommen.[9]

Sie sollten rote Texte nicht ignorieren, aber sich auch nicht verunsichern lassen. Achten Sie also auf rote Texte! Auch wenn es sich nicht immer um ein Problem handelt, helfen Warnungen häufig bei der Behebung und Vermeidung von Fehlern. Ein wesentlicher Teil der Entwicklungsarbeit besteht in der Bearbeitung und Lösung von solchen Problemen.

5.1.3 Grundlagen der Programmierung

Datentypen In vielen Programmierumgebungen lassen sich grundlegend zwei Datensorten unterscheiden, einfache und zusammengesetzte Werte. **Einfache Datentypen** bilden den Grundstock für die Programmierung und werden auch Skalare,[10] elementare oder atomare Objekte genannt. Hierzu zählen insbesondere verschiedene Arten von Zahlen, Wahrheitswerte und Zeichenketten. Bei Zeichenketten, also Text, ist zu beachten, dass sie in Anführungszeichen gesetzt werden. Andernfalls wäre nicht klar, ob nicht doch der Name eines Objekts statt der Zeichenkette gemeint ist (Tab. 5.1). Einige Werte haben besondere Bedeutungen, wichtig ist vor allem die Kennzeichnung fehlender Werte durch den Ausdruck `NA` (Tab. 5.2).

Darüber hinaus gibt es in R mit `factor` einen besonderen Typ für kategoriale Daten. Statt etwa zur Kennzeichnung der Haarfarbe einer Person nur Zahlen

Tab. 5.1 Einfache Datentypen in R

Datentyp	Ausdruck	Beispiel
Ganze Zahlen	integer	`1884`
Kommazahlen	double	`1.6`
Wahrheitswerte	logical	`TRUE, FALSE`
Zeichenketten	character	`"Anna"`

Quelle: Eigene Darstellung

[8] Siehe Stack Overflow (2022; https://stackoverflow.com/).

[9] Eine Auflistung weiterer häufiger Fehlerquellen findet sich in Abschn. 1.3.

[10] Genau genommen gibt es in R keine Skalare, sondern nur atomare Objekte. Einem der Entwickler von R wird die Äußerung zugeschrieben: „Everything that exists is an object. Everything that happens is a function call“ (Chambers 2014, S. 170).

Tab. 5.2 Werte mit besonderen Bedeutungen in R

Datentyp	Ausdruck
Fehlende Werte	`NA`
Unendlich	`Inf, -Inf`
Keine Zahl	`NaN`
Kein Wert	`NULL`

Quelle: Eigene Darstellung

Tab. 5.3 Vektoren und Listen in R

Datentyp	Ausdruck	Beispiel
Vektor	`c`	`c(1, 5, 9)`
Liste	`list`	`list(1, 5, 9)`

Quelle: Eigene Darstellung

(Dummy-Kodierung) oder nur Zeichenketten (die Namen der Farben) zu verwenden, verbinden Faktoren beides. Jede Haarfarbe bzw. jede Ausprägung einer Variablen erhält sowohl eine Nummer als auch eine Bezeichnung. Für die Ausgabe zum Beispiel in Grafiken werden die Bezeichnungen verwendet, die Reihenfolge der Kategorien wird dagegen über die Nummer bestimmt.

Neben einzelnen Werten können mehrere elementare Werte in zusammengesetzten Datentypen gespeichert werden. Die einfachste Form solcher Datentypen sind Vektoren und Listen (Tab. 5.3). Diese beiden Strukturen sind in R sehr häufig anzutreffen, denn viele Funktionen können nicht nur einzelne Werte verarbeiten, sondern die Funktion gleich auf alle Werte eines Vektors anwenden. Diese vektorisierten Funktionen sind ein Unterschied zu vielen anderen Programmiersprachen und besonders vorteilhaft für effiziente Datenanalysen.[11]

Im ersten Moment scheinen Listen und Vektoren das Gleiche zu sein, sie unterscheiden sich aber in einem wichtigen Punkt. In einem Vektor können nur Elemente mit dem gleichen Typ enthalten sein, also entweder nur Zahlen oder nur Zeichenketten, während in Listen diese verschiedenen Typen gleichzeitig möglich sind. Mit der `c()`-Funktion lassen sich Vektoren nicht nur erstellen, sondern auch um weitere Elemente erweitern:

```
personen <- c("Bea", "Leo")
personen <- c(personen, "Niska")
print(personen)
```

[11] In anderen Programmiersprachen werden für den gleichen Zweck häufiger Schleifen (foreach, for in etc.) und ähnliche Kontrollstrukturen eingesetzt.

Die Anzahl der Elemente in einem Vektor oder einer Liste kann mit der `length()`-Funktion ermittelt werden. Elemente in Listen und Vektoren können auch benannt werden, zum Beispiel:

```
person <- list(vorname="Inger", nachname="Engmann")
```

Ein weiterer komplexer Datentyp sind **tabellenartige Datenstrukturen,** welche in der Datenanalyse häufig verwendet werden. Die Standardstruktur für Datensätze ist in R der Typ `data.frame`, der aus Spalten und Zeilen besteht. Ein Dataframe ist intern als Liste von Vektoren organisiert. Jede Spalte ist ein Vektor mit einem einheitlichen Typ und die gesamte Tabelle besteht somit aus einer Liste dieser Vektoren. Die Spalten haben Namen und auch die Zeilen könnten benannt werden, sie werden aber normalerweise einfach durchnummeriert.

```
personen <- data.frame(
  name  = c("Bea", "Leo", "Niska", "Inger", "Tobbe"),
  alter = c(24, 26, 25, 52, 17),
  typ   = c("Bot", "Bot", "Bot", "Mensch", "Mensch")
)
```

In RStudio kann der Dataframe `personen` anschließend als Tabelle angezeigt werden (Abb. 5.5).

Es gibt verschiedene Möglichkeiten, um auf Teile eines Dataframes zuzugreifen. Spalten, Zeilen und Zellen können zunächst über eckige Klammern `[]` ausgewählt werden, die an den Namen des Dataframe angehängt werden. Der erste

	name	alter	typ
1	Bea	24	Bot
2	Leo	26	Bot
3	Niska	25	Bot
4	Inger	52	Mensch
5	Tobbe	17	Mensch

Abb. 5.5 Dataframe in RStudio. (Quelle: eigene Darstellung)

Wert in der Klammer gibt den Zeilenindex an, nach einem Komma folgt die gewünschte Spalte. Wird der Zeilen- oder Spaltenindex leer gelassen, werden alle Werte in der entsprechenden Spalte bzw. Zeile aufgerufen. Tippen Sie die folgenden Befehle in das Skript ein und führen Sie es aus, um die Funktionsweise zu verstehen

- Alle Werte der ersten Zeile werden ausgewählt, zurückgegeben wird ein Dataframe mit allen Spalten: `personen[1, ]`
- Alle Werte der ersten Spalte werden ausgewählt; da es sich nur um eine einzige Spalte handelt, wird ein Vektor zurückgegeben: `personen[, 1]`
- Um einen einzelnen Wert auszuwählen, wird die Auswahl der Zeilen und Spalten kombiniert: `personen[1, 1]`

Sollen mehrere Spalten oder Zeilen ausgewählt werden, dann wird an den entsprechenden Stellen ein Vektor mit den Namen oder Nummern von Spalten bzw. Zeilen übergeben werden. Mit einem Doppelpunkt werden Bereiche festgelegt:

- Auswahl mehrerer Spalten: `personen[, c("name", "alter")]`
- Auswahl mehrerer Zeilen: `personen[c(1:3), ]`

Für die Auswahl von Zeilen ist es häufig gewünscht, eine Auswahl nach festgelegten Kriterien vorzunehmen. Die Kriterien werden dann vor dem Komma formuliert:

```
personen[personen$alter > 30, ]
```

Weitere Möglichkeiten zum Formulieren von Bedingungen finden Sie in Tab. 5.4. Bei der Formulierung der Bedingung kommt im Beispiel eine weitverbreitete Technik zum Adressieren einzelner Spalten zum Einsatz. Einzelne Spalten können mit dem Dollarzeichen gefolgt vom Spaltennamen ausgegeben werden:

```
personen$name
```

Alternativ können Spalten über ihre Nummer, die in doppelten eckigen Klammern angegeben wird, erreicht werden:

```
personen[[1]]
```

Tab. 5.4 Wahrheitsbedingungen in R

<table>
<tr><th>Beispiel</th><th>Erläuterung</th></tr>
<tr><td><code>typ == "Bot"</code></td><td>Gleichheit</td></tr>
<tr><td><code>typ != "Bot"</code></td><td>Ungleichheit</td></tr>
<tr><td><code>alter > 20</code> bzw. <code>alter >= 20</code></td><td>Größer bzw. Größergleich</td></tr>
<tr><td><code>alter < 20</code> bzw. <code>alter <= 20</code></td><td>Kleiner bzw. Kleinergleich</td></tr>
<tr><td><code>is.na(hashtags)</code></td><td>Prüfen, ob ein Wert fehlt</td></tr>
<tr><td><code>!is.na(hashtags)</code></td><td>Mit dem Negator <code>!</code> prüfen, ob ein Wert nicht fehlt</td></tr>
<tr><td><code>(typ == "Bot") &(alter < 20)</code></td><td>Mit dem Operator <code>&</code> Ausdrücke über Und-Bedingungen verknüpfen</td></tr>
<tr><td><code>(alter < 20) | (alter > 30)</code></td><td>Mit dem Operator <code>|</code> Ausdrücke über Oder-Bedingungen verknüpfen</td></tr>
</table>

Beachten Sie, dass analog zu den Beispielen im Text jeweils der Datensatz ergänzt werden muss. (Quelle: Eigene Darstellung)

Diese Notation mit doppelten eckigen Klammern oder dem Dollarzeichen funktioniert für alle Listen. Da ein Dataframe eine Liste aus Vektoren ist, wird im Beispiel das erste Element der Liste, das heißt der erste Vektor, angesprochen.

Auf die gleiche Weise, das heißt aufbauend auf die Auswahl von Zeilen, Spalten und Zellen, können auch Werte verändert werden. So würde mit folgendem Beispiel die Namensspalte für alle Zeilen auf den Wert „anonym“ geändert werden:

```
personen$name <- "anonym"
```

Es wurde nur ein Wert zugewiesen, obwohl mehrere Zeilen betroffen sind. In diesem Fall wird der angegebenen Wert recycled und auf alle Zeilen übertragen. Stattdessen kann in einem Vektor für jede Zeile ein eigener Wert festgelegt werden:

```
personen$name <-
  c(
     "Bea", "Leo", "Niska",
     "Inger", "Tobbe"
   )
```

Wenn die Spalte noch nicht existiert, wird sie automatisch neu angelegt:

```
personen$nummer <- c(1, 2, 3, 4, 5)
```

Um eine Spalte wieder zu löschen, wird der Wert `NULL` zugewiesen:

```
personen$nummer <- NULL
```

Diese Grundtechniken gehören zur Basisausstattung von R. Prägen Sie sich die Möglichkeiten zum Umgang mit Zeilen und Spalten ein, schlagen Sie immer wieder nach und kombinieren Sie die Befehle, um nach und nach eigene Routinen zu entwickeln.

Neben den Dataframes kommen in einigen Fällen zwei ganz ähnliche, komplexe Datentypen zum Einsatz. Erstens: Wenn in allen Spalten einer Tabelle der gleiche Datentyp enthalten ist – zum Beispiel nur Zahlen –, dann kann dafür auch der Datentyp `matrix` eingesetzt werden. Matrizen haben den Vorteil, dass sie sich schnell umformen lassen und einige spezielle Rechenoperationen erlauben (siehe Abschn. 4.2.4). Zweitens: Aus dem Package *tibble* des Tidyverse-Universiums stammt der Datentyp `tbl_df`. Dieser wird auch als Tibble bezeichnet und ist in den meisten Fällen eine etwas komfortablere Variante von Dataframes (siehe Abschn. 5.1.4). Beispielsweise ist die Ausgabe über den `print()`-Befehl kompakter.

Zu Beginn ist die Vielfalt der Datentypen verwirrend, aber mit der Zeit werden Sie lernen, für welche unterschiedlichen Anwendungsfälle sie sich eignen. Der Typ eines Objekts kann mit der `str()`-Funktion festgestellt werden:

```
str(personen)
```

Diese Funktion hilft immer dann weiter, wenn man es mit unbekannten Objekten zu tun, etwas nicht funktioniert oder wenn man den Überblick verloren hat.

Viele Typen lassen sich ineinander umwandeln, etwa um eine Zeichenkette in eine echte Zahl zu konvertieren:

```
as.numeric("1852")
```

Umwandlungsfunktionen beginnen in R in der Regel mit `as`, sodass beim Eintippen von „as“ im Skript automatisch Funktionen vorgeschlagen werden. Mithilfe entsprechender Funktionen können auch Dataframes, Matrizen und Tibbles ineinander überführt werden:

```
as.matrix(personen)
as.data.frame(personen)
as_tibble(personen)
```

Loops und andere Kontrollstrukturen Wenn Befehle auf mehrere Werte angewendet werden, kommen Kontrollstrukturen zum Einsatz – etwa wenn eine Liste oder ein Datensatz mit Personen vorliegt und man für jede Person einzeln das Alter bestimmen will. Kontrollstrukturen legen fest, wie und auf welchen Daten die einzelnen Befehle auszuführen sind.

Mithilfe von **For-Schleifen** (engl. *for-loop*) kann man Funktionen auf die einzelnen Einträge einer Liste anwenden:

```
for (item in personen$name) {
  message <- paste0(item, " is a bot")
  print(message)
}
```

Im Beispiel wird die Liste aller Personennamen verwendet. Innerhalb der `for()`-Anweisung wird diese Liste hinter dem Ausdruck `in` angegeben. Die Liste enthält die einzelnen Einträge, für diese Einträge wird der in geschweiften Klammern angegebene Codeblock jeweils einmal ausgeführt. Innerhalb des Codeblocks steht der jeweilige Eintrag dann im Objekt `item` zur Verfügung. Die Bezeichnung `item` kann man beliebig austauschen; die Schleife würde ebenso mit folgender Formulierung funktionieren:

```
for(name in personen$name) {
  message <- paste0(name, " is a bot")
  print(message)
}
```

Diese beiden For-Schleifen erzeugen dementsprechend beide den gleichen Output:

```
[1] "Bea is a bot"
[1] "Leo is a bot"
[1] "Niska is a bot"
[1] "Inger is a bot"
[1] "Tobbe is a bot"
```

Im Beispiel ist nun jedoch nicht jede Person auch tatsächlich ein Bot. Aus der Spalte `typ` lässt sich entnehmen, dass einige Personen Menschen sind. Um das zu berücksichtigen, kann man den Programmablauf durch **If-Bedingungen** verzwei-

gen. Dadurch werden die Funktionen nur dann ausgeführt, wenn eine bestimmte Bedingung erfüllt ist – etwa nur, wenn in der Spalte `typ` der Eintrag „Bot“ steht.

Dafür benötigt man innerhalb der Spalte nicht nur den Namen einer Person, sondern auch deren Typ. Insofern reicht es nicht aus, über eine einzelne Liste zu loopen. Stattdessen kann man die Zeilen eines Datensatzes durchlaufen, indem man über eine Liste der Zeilenindizes (das sind die Nummern, mit denen eine Zeile adressierbar ist) iteriert. Mit dem Ausdruck `1:nrows(personen)` lässt sich eine solche Liste erzeugen.[12] Innerhalb des Codeblocks kann dann die betroffene Zeile wie oben beschrieben zur Weiterverarbeitung ausgewählt werden (Abschn. 5.1.3):

```
for (i in 1:nrow(personen)) {
  row = personen[i, ]

  if (row$typ == "Bot") {
    message <- paste0(row$name, " is a bot")
    print(message)
  }
}
```

Die einzelnen Zeilen liegen dann im Objekt `row` vor. Für jede einzelne Zeile lässt sich schließlich entscheiden, ob in dieser Zeile das Merkmal „Bot“ enthalten ist. If-Anweisungen enthalten eine Wahrheitsbedingung, die hier über den Vergleichsoperator == erstellt wird. Ist die Bedingung erfüllt, wird eine Meldung ausgegeben. Anderenfalls geschieht nichts. Solche Wahrheitsbedingungen (engl. *boolean expressions*) werden ebenso für das Filtern von Datensätzen benötigt (Abschn. 5.1.4) und können miteinander kombiniert werden, wobei es sich empfiehlt, Teilausdrücke in Klammern anzugeben (Tab. 5.4).

Eine Erweiterung der If-Bedingung stellt die **If-else-Verzweigung** dar. Der Else-Block wird immer dann ausgeführt, wenn die If-Bedingung *nicht* erfüllt ist:

```
if (row$typ == "Bot") {
  message <- paste0(row$name, " is a bot")
} else {
  message <- paste0(row$name, " is a human")
}

print(message)
```

[12] Die Funktion `nrows()` bestimmt die Anzahl der Zeilen des Dataframe `personen`. Im Beispieldatensatz sind das fünf Zeilen. Durch den Doppelpunkt wird ein Bereich markiert, also der Bereich von eins bis fünf: 1, 2, 3, 4, 5.

Für komplexere Anwendungsfälle lassen sich beliebig viele If-else-Anweisungen aneinanderhängen, wobei schrittweise die Bedingungen durchgeprüft und dann nur der erste passende Block ausgeführt wird:

```
if (row$alter < 20) {
  print("A young person")
}
else if (row$alter < 30) {
  print("A twen")
}
else {
  print("How old is old?")
}
```

Schleifen, Bedingungen und Verzweigungen sind die Basiswerkzeuge der meisten Programmiersprachen. Charakteristisch für R sind darüber hinaus vektorisierte Funktionen, die auch ohne Schleifen eine Liste von Werten verarbeiten können. Viele Basisfunktionen sind bereits vektorisiert, so loopt der `paste0()`-Befehl automatisch über die Liste der Personennamen und hängt an jeden Eintrag eine Zeichenkette an:

```
paste0(personen$name, " is alive.")
```

Ist eine Funktion noch nicht vektorisiert, kann sie über `lapply()` (abgekürzt von engl. *list apply*) auf jedes einzelne Element angewendet werden. Als ersten Parameter gibt man die Liste an, der zweite Parameter enthält den Funktionsnamen. Beim Ausführen wird jeweils ein Eintrag der Liste als erster Parameter dieser Funktion verwendet. Angaben an späteren Stellen werden als weitere Parameter an die Funktion übergeben:

```
lapply(personen$name, paste0, " is a bot")
```

Da `paste0()` bereits vektorisiert ist, leistet dieser Aufruf im Prinzip das Gleiche wie die direkte Verwendung von `paste0()`. Einen Unterschied gibt es allerdings: `lapply()` gibt selbst auch wieder eine Liste zurück. Vergleichen Sie die Outputs! Und auch wenn es im Beispiel noch nicht nötig wäre, können Sie sich dieses Muster bereits jetzt merken. Denn Vektorisierung erhöht meistens die Effizienz von Code – Berechnungen lassen sich intern beschleunigen und das Skript wird kompakter.

5.1.4 Grundlagen der Datenanalyse

Eine besondere Sammlung von Funktionen für die Aufbereitung und Analyse von Daten ist das Tidyverse.[13] Zum Tidyverse gehört eine Reihe von Packages, deren Funktionen im Folgenden eingeführt werden:

- *tibble* führt einen Datentyp `tbl_df` ein, der die Dataframes von R verbessert (Müller et al. 2022b).
- *readr, readxl* und *writexl* erleichtern das Einlesen und Speichern von Datensätzen mit Funktionen wie `read_csv()` und `read_xlsx()` (Wickham et al. 2022b, e; Ooms und McNamara 2021).
- *dplyr* stellt Funktionen für das Filtern, Zusammenführen und Aggregieren von Datensätzen bereit, als Alternative zu den oben besprochenen Selektionsbefehlen (Wickham et al. 2022a).
- *tidyr* dient der Umformung von Datensätzen zwischen Wide- und Long-Format (Wickham und Girlich 2022; siehe Abschn. 4.2.4).
- *skimr* erlaubt mit der `skim()`-Funktion einen schnellen Überblick über die Variablen eines Datensatzes (Waring et al. 2022).
- *ggplot2* stellt eine Vielzahl mächtiger Funktionen zum Erzeugen von Grafiken bereit (Wickham et al. 2022c).

Weitere nützliche Packages aus dem Tidyverse beziehen sich auf spezielle Datentypen wie *stringr* für Zeichenketten sowie reguläre Ausdrücke (Wickham 2019a; siehe Abschn. 4.1.1), *lubridate* für Datums- und Zeitangaben (Spinu et al. 2021) und *DBI* für den Zugriff auf Datenbanken wie MySQL (Müller et al. 2022a; siehe Abschn. 3.3).

Das wichtigste Konzept des Tidyverse besteht darin, die Welt durch eine viereckige Brille zu betrachten: Alle Daten werden möglichst in der Form von Tibbles verarbeitet. Tibbles funktionieren wie ganz normale Tabellen, für die gilt: Jede Beobachtungseinheit wird in einer Tabelle erfasst, jede Variable in einer Spalte, jede Beobachtung in einer Zeile und jeder Wert in einer Zelle (siehe Abschn. 4.2). Bei der Verwendung von Tidyverse-Funktionen ersetzen Tibbles automatisch die in R eingebauten Dataframes. Dadurch sind die verschiedenen Funktionen des Tidyverse häufig untereinander besser kompatibel als die R-Basisfunktionen und besonders für die Datenaufbereitung zu empfehlen. Nicht immer ist die Arbeit mit dem Tidyverse allerdings übersichtlicher oder schneller als mit den Basis-R-Funktionen, deshalb sollte man sich in beide Bereiche einarbeiten.

[13] Siehe Wickham (2019b; https://www.tidyverse.org).

	id	from	favorites	replies	retweets	media
1	1	eaduenergy	0	0	NA	text
2	2	eaduenergy	9	0	8	text
3	3	eaduenergy	6	0	NA	image

	id	from	media	metric	value
1	1	eaduenergy	text	favorites	0
2	1	eaduenergy	text	replies	0
3	1	eaduenergy	text	retweets	NA
4	2	eaduenergy	text	favorites	9
5	2	eaduenergy	text	replies	0
6	2	eaduenergy	text	retweets	8
7	3	eaduenergy	image	favorites	6
8	3	eaduenergy	image	replies	0
9	3	eaduenergy	image	retweets	NA

Abb. 5.6 Umformen vom Wide- ins Long-Format. Fantasiedaten nur zu Demonstrationszwecken. (Quelle: eigene Darstellung)

Datensätze einlesen und abspeichern Die bisherigen Beispiele bezogen sich auf einen sehr kleinen Datensatz, der als Dataframe direkt im Skript erstellt wurde. In der Regel liegen die Daten aber in eigenen Dateien vor, zum Beispiel in CSV-Dateien. Mit dem CSV-Format können nahezu alle Programme umgehen, die zur Erfassung oder Analyse von Daten eingesetzt werden (siehe Abschn. 3.3). Die Beispiele der folgenden Kapitel beziehen sich auf den in Abb. 5.6 dargestellten Datensatz (☛ *Repositorium*).

Das Einlesen und Speichern von CSV-Dateien gehört zu den Basisfunktionen von R, ein Beispiel ist die Funktion `read.csv()`. Günstiger sind aber in der Regel die Funktionen aus dem Tidyverse, die mit einem Unterstrich statt Punkt geschrieben werden. Beispiele sind aus dem *readr*-Package die Funktionen `read_csv()` sowie `read_csv2()`. Dieses Package wird automatisch geladen, wenn man das *tidyverse*-Package lädt. Sind die Daten mit einem Komma getrennt (engl. *comma separated values*), wird die Funktion `read_csv()` verwendet. Mit Semikolon getrennte Daten liest die Funktion `read_csv2()` ein:

```
library(tidyverse)
tweets <- read_csv2("example-tweets.csv")
```

Diese Funktion nimmt als ersten Parameter den Dateinamen in Anführungszeichen entgegen und gibt den Datensatz als Tibble zurück. Je nach Aufbau der Datei werden andere Funktionen oder weitere Parameter benötigt. Wenn die Daten mit einem Tabulator getrennt sind (engl. *tab separated values*), dann hilft die Funktion `read_tsv()`. Die Funktion `read_delim()` lässt sich umfangreich konfigurieren, um weitere Varianten von CSV-Formaten zu berücksichtigen. Der folgende Aufruf wäre identisch mit `read_csv2()`, da ein Semikolon als Trennzeichen definiert wird:

```
tweets <- read_delim("example-tweets.csv", delim=";")
```

Die Funktionen im Tidyverse haben meistens passendere Voreinstellungen als die Basisfunktionen, beispielsweise behandeln sie Text als Text und nicht als Faktor. Analog stehen Funktionen zum Speichern von CSV-Dateien zur Verfügung. Als erster Parameter wird dann der Datensatz angegeben und der zweite Parameter bestimmt den Dateinamen:

```
write_csv(tweets, "tweets.csv")
```

Wenn man konsequent mit Funktionen aus dem Tidyverse arbeitet, dann lassen sich die meisten Ergebnisse als Tabellen abspeichern, zum Beispiel das Ergebnis der unten besprochenen `count()`-Funktion. Im folgenden Beispiel wird das Ergebnis zunächst in einem eigenen Objekt abgelegt und dann als CSV-Datei abgespeichert:

```
tweets_jeautor <- count(tweets, from)
write_csv(tweets_jeautor, "tweets_jeautor.csv")
```

Auf diese Weise lässt sich der Output der Datenanalyse gut dokumentieren und weiterverwenden. Auch beim Abspeichern sind verschiedene CSV-Varianten möglich. Die folgende Funktion verwendet als Trennzeichen ein Semikolon, als Dezimalzeichen das Komma und markiert die Zeichenkodierung als UTF-8 (mit einer Byte Order Mark, siehe Abschn. 3.3). Diese Voreinstellungen sind besonders für den Austausch mit Excel geeignet:

```
write_excel_csv2(
  tweets_jeautor,
  "tweets_jeautor.csv"
)
```

Für den Austausch mit Excel gibt es in den Paketen *readxl* und *writexl* auch eigene Funktionen:[14]

```
write_xlsx(tweets_jeautor, "tweets_jeautor.xlsx")

tweets_jeautor <- read_xlsx("tweets_jeautor.xlsx")
```

Die Dateienamen beziehen sich in allen Fällen auf das aktuelle Arbeitsverzeichnis. Wenn man mit RStudio ein Projekt angelegt hat, dann ist das Arbeitsverzeichnis zu Beginn immer das Projektverzeichnis. Es können zum Einlesen der Dateien und zum Speichern relative Pfade verwendet werden, durch die auf Unterordner im Arbeitsverzeichnis verwiesen wird. Ebenso kann durch die Angabe `../` zu Beginn des Pfades eine Ordnerebene aufwärts navigiert werden. Es empfiehlt sich, Daten und Ergebnisse in Unterverzeichnissen abzulegen, zum Beispiel in einem Unterverzeichnis mit dem Namen „daten“:

```
write_xlsx(tweets, "daten/example-tweets.xlsx")

tweets <- read_xlsx("daten/example-tweets.xlsx")
```

Datenaufbereitung mit dem Tidyverse Liegen die Daten in einem Tibble vor, können sie mit `select()` bzw. `filter()` auf die relevanten Spalten bzw. Zeilen zugeschnitten werden. Beide Funktionen erwarten als erstes Argument ein Tibble und geben ein Tibble zurück. Die folgende Anweisung wählt alle Zeilen aus, bei denen in der Spalte `from` der Wert „unialdera“ steht, und legt das Ergebnis im Objekt `auswahl` ab:

```
auswahl <- filter(tweets, from == "unialdera")
```

[14] Für die Arbeit mit Textkorpora müssen manchmal Textdateien oder auch Worddokumente eingelesen werden – Beispiele dafür finden Sie in Kap. 9. Bei Dateien aus anderen Statistikprogrammen wie SPSS, Stata oder SAS hilft das Paket *haven* weiter (Wickham, Miller & Smith 2022).

Derartige Filterbedingungen können mit den gleichen Wahrheitsausdrücken wie bei If-Bedingungen formuliert werden (siehe Abschn. 5.1.3). Eine Besonderheit im Tidyverse besteht darin, dass Spaltennamen immer ohne Anführungszeichen und ohne Bezug auf das Tibble angegeben werden. In den Basisfunktionen wäre dagegen die Angabe `tweets$from == "unialdera"` nötig.

Eine Auswahl von Spalten wird mit `select()` erreicht, die gewünschten Spalten wie beispielsweise `from` und `favorites` werden hinter dem Tibble aufgezählt:

```
auswahl <- select(tweets, from, favorites)
```

Die Funktionen aus dem Tidyverse sind meistens ähnlich aufgebaut, verlangen fast immer als ersten Parameter ein Tibble und geben zudem ein Tibble zurück. Dadurch lassen sich verschiedene Funktionen mit dem Pipe-Operator `%>%` aneinanderketten. Da dieser Operator häufig eingesetzt wird, können Sie sich die Tastenkombination dafür angewöhnen (Mac: Command + Shift + M; Windows: Strg + Shift + M). Der Pipe-Operator schiebt das vorangegangene Objekt in den ersten Parameter einer Funktion, sodass die folgenden beiden Aufrufe identisch sind:

```
select(tweets, from, favorites)

tweets %>% select(from, favorites)
```

Die Pipe wird dann nützlich, wenn mehrere Aufbereitungsschritte nacheinander ausgeführt werden sollen:

```
reactions <- tweets %>%
  filter(media == "image") %>%
  mutate(react = favorites + replies + retweets) %>%
  select(from, react) %>%
  arrange(-react)
```

Jedem Befehl hinter einer Pipe wird als erster Parameter automatisch das vorherige Ergebnis übergeben. So werden in dem Beispiel zunächst alle Zeilen des Datensatzes `tweets` herausgefiltert, in denen in der `media`-Spalte der Wert „image" steht. Sodann wird mit `mutate()` eine neue Spalte `react` erstellt, in der die verschiedenen Reaktionen je Tweet summiert werden. Im nächsten Schritt werden aus dem gefilterten und ergänzten Tibble die Spalten `from` und `react`

Tab. 5.5 Datenaufbereitungsfunktionen aus dem Tidyverse (Auswahl)

Funktion	Beschreibung
`filter()`	Auswählen von Zeilen in Datensätzen
`select()`	Auswählen oder Umbenennen von Spalten
`rename()`	Umbenennen von Spalten
`arrange()`	Sortieren von Zeilen in Datensätzen
`mutate()`	Neuerstellen von Spalten
`count()`	Zählen von Zeilen, ggf. nach den Werten in ausgewählten Spalten
`group_by()`	Gruppieren von Zeilen in Datensätzen
`ungroup()`	Auflösen einer Gruppierung
`summarize()`	Aggregieren von Zeilen, zum Beispiel zur Berechnung von Summen oder zur Bestimmung der Anzahl innerhalb einer Gruppe
`slice_sample()`	Ziehen einer Zufallsauswahl
`slice_max()`	Auswählen von Zeilen mit den größten Werten in einer ausgewählten Spalte
`pivot_longer()`	Umformen vom Wide- in das Long-Format (*tidyr*-Package)
`pivot_wider()`	Umformen vom Long- in das Wide-Format (*tidyr*-Package)

Sofern nicht anders angegeben, stammen die Funktionen aus dem *dplyr*-Package. Quelle: Eigene Darstellung

ausgewählt. Schließlich wird mit `arrange()` sortiert, das Minuszeichen sorgt für eine absteigende Sortierung der Spalte `react`. Abgespeichert wird das finale Tibble unter der Bezeichnung `reactions`. Mit der Pipe wird der erste Parameter also jeweils übersprungen, da er aus der Zeile vorher stammt. Diese Aneinanderkettung funktioniert insbesondere mit den Funktionen aus dem Package *dplyr* sehr gut (Tab. 5.5) und spart Zuweisungen ein, was für eine bessere Übersichtlichkeit sorgt.

Manchmal müssen Daten vom Wide- in das Long-Format oder umgekehrt umgeformt werden (siehe Abschn. 4.2.1). So liegen im Beispieldatensatz (☛ *Repositorium*) Reaktionen auf einen Tweet in den Spalten `favorites`, `replies` und `retweets` vor. Für einige Analysen kann es hilfreich sein, alle Reaktionen in einer einzigen Spalte zu erfassen. Die nebeneinanderstehenden (= wide) Spalten sollen also zu untereinanderstehenden (= long) Zeilen umgeschichtet werden. Dafür kommt die Funktion `pivot_longer()` aus dem *tidyr*-Package zum Einsatz. Dem Parameter `cols` gibt man eine Liste der Spalten mit, die zusammengefasst werden. Im Parameter `names_to` wird festgelegt, wie die Spalte mit den ursprünglichen Spaltennamen heißen soll. Wenn nicht anders festgelegt, landet die Anzahl der jeweiligen Reaktionsart in der `value`-Spalte:

```
tweets_long <- tweets %>%
  pivot_longer(
    cols = c(favorites, replies, retweets),
    names_to = "metric",
    values_to = "value"
  )
```

Das neue Tibble `tweets_long` enthält nun dreimal mehr Zeilen als das alte Tibble `tweets`, da nun alle Reaktionen untereinander aufgeführt sind (Abb. 5.6). Diese Umstrukturierung ermöglicht es beispielsweise, die Mittelwerte für alle Reaktionsarten auf einmal auszurechnen (siehe unten).

Funktionen für die Datenanalyse Am Anfang der Datenanalyse steht in der Regel die Beschreibung des Datensatzes. Mit der `count()`-Funktion lässt sich zunächst die Anzahl der Fälle auszählen:

```
tweets %>% count()
```

Wenn der Datensatz in einer Spalte kategoriale Merkmale wie Namen enthält, lässt sich die Fallzahl für jede einzelne Gruppe ermitteln:

```
tweets %>% count(from)
```

Damit wird ein typisches Szenario der Datenanalyse umgesetzt, das als Split-Apply-Combine bezeichnet wird (siehe Abschn. 4.2.4). Ein Datensatz wird zunächst in Teilgruppen aufgeteilt, dann wird auf diese Teilgruppen eine Funktion angewendet und schließlich wird das Ergebnis zusammengefügt. Das gleiche Ergebnis lässt sich mit einer Kombination aus `group_by()` zum Aufteilen der Gruppen, `summarize()` für die Zusammenfassung jeder Gruppe und schließlich `ungroup()` zum Auflösen der Gruppierung erzielen:

```
tweets %>%
   group_by(from) %>%
   summarize(n = n()) %>%
   ungroup()
```

Verwirrend ist möglicherweise das doppelte Vorkommen von `n` innerhalb der `summarize()`-Funktion. Vor dem Gleichheitszeichen wird der Name der Spalte

angegeben, in der das Ergebnis gespeichert werden soll. Sie können diese einfach umbenennen, z. B. zu `anzahl`. Nach dem Gleichheitszeichen wird die Funktion `n()` mit leeren Klammern aufgerufen, das heißt ohne Argumente. Diese Funktion zählt die Anzahl der Zeilen innerhalb der Gruppe.

Im nächsten Schritt interessieren meist die Zusammenhänge zwischen mehreren Variablen. Die Kombination von mehreren kategorialen Variablen kann ebenfalls mit `count()` ausgezählt werden, etwa um die Anzahl der Bilder, Links und Texte (Spalte `media`) verschiedener Accounts (Spalte `from`) zu vergleichen:

```
tweets %>%
  count(from, media)
```

Daraus wird eine Kreuztabelle mit den `media`-Werten als Spalten, wenn man das Ergebnis in das Wide-Format umformt – probieren Sie es aus und ändern Sie dabei den Parameter `names_from = media` auch einmal auf `from`, um das Prinzip zu verstehen:

```
tweets %>%
  count(from, media) %>%
  pivot_wider(names_from = media, values_from = n)
```

Liegen dagegen metrische Daten wie die Anzahl der Favorites oder Altersangaben vor, beschreibt man die Verteilungen durch Kennwerte wie den Mittelwert und die Standardabweichung, Minimum und Maximum oder Quartile. Der Mittelwert kann mit der R-Basisfunktion `mean()` ausgerechnet werden:

```
mean(tweets$favorites)
```

Um mehrere Gruppen hinsichtlich metrischer Merkmale zu vergleichen, kann wiederum `summarize()` verwendet werden:

```
tweets %>%
  group_by(from) %>%
  summarize(favs = mean(favorites)) %>%
  ungroup()
```

Liegen die Daten im Long-Format vor (siehe oben), kann der Mittelwert gleich für mehrere Reaktionsarten ausgerechnet werden. Mit dem Parameter `na.rm = T` werden dabei fehlende Werte ignoriert:

```
tweets_long %>%
  group_by(metric) %>%
  summarize(m = mean(value, na.rm = T)) %>%
  ungroup()
```

Zu beachten ist bei der Verwendung von `summarize()`, dass im Ergebnis nur die Gruppierungsvariablen und die innerhalb der Funktion neu erstellten Spalten übrig bleiben. Auch wenn innerhalb von `summarize()` im Prinzip mehrere neue Variablen gleichzeitig berechnet werden könnten, ist es etwas mühsam, auf diese Weise für mehrere Variablen verschiedene Kennwerte wie Mittelwerte, Standardabweichungen und Quartile zu berechnen. Einen schnelleren Überblick über die Variablen eines Datensatzes erhält man mit der Funktion `skim()` aus dem *skimr*-Package:

```
library(skimr)
tweets %>% skim(favorites)
```

Um Kategorien hinsichtlich metrischer Merkmale zu vergleichen, wird `skim()` mit `group_by()` kombiniert:

```
tweets %>%
  group_by(media) %>%
  skim(favorites, replies, retweets) %>%
  ungroup()
```

Für den Zusammenhang zwischen zwei metrischen Merkmalen wird schließlich auf die Korrelationsanalyse zurückgegriffen:

```
cor(tweets$favorites, tweets$replies)
```

Die aufgeführten Befehle gehören zum Standardrepertoire der deskriptiven Datenanalyse. Darüber hinaus ist das Basiswerkzeug für die statistische Analyse metrischer Merkmale die lineare Regression.[15] Ein lineares Regressionsmodell – wenn man den statistischen Zusammenhang zwischen abhängigen und unabhängigen Va-

[15] Mit Regressionsanalysen wird der Einfluss der unabhängigen Variablen auf die abhängigen Variablen modelliert, etwa der Einfluss von Alkoholkonsum auf die Leistungsfähigkeit. Eine umfassende Einführung findet sich in Field, Miles & Field (2012). Spezifische Regressionsmodelle können auch für kategoriale Daten verwendet werden, siehe Abschn. 8.1.

riablen prüfen will – wird über die Funktion `lm()` erstellt. Die Funktion nimmt zunächst die abhängige Variable entgegen, gefolgt von einer Tilde ~ und dann der unabhängigen Variable oder einer Kombination mehrerer unabhängiger Variablen. Über den Parameter `data` wird der Datensatz mitgegeben. Der Output der Funktion kann in einem beliebigen Objekt abgelegt werden – im Beispiel `fit`. Das Auftreten von Favorites in Abhängigkeit der Retweets wird beispielsweise bestimmt über:

```
fit <- lm(favorites ~ retweets, data = tweets)
```

Über `summary(fit)` werden dann gängige Kennwerte der Regressionsanalyse wie der Interzept, die Gewichtungsfaktoren und Signifikanzmaße ausgegeben (Abb. 5.7).

R bringt eine Vielzahl weiterer Analysefunktionen, statistischer Tests und Modellierungswerkzeuge mit. Versuchen Sie herauszufinden, wie die Funktionen in Tab. 5.6 funktionieren: Geben Sie die Namen in ein R-Skript ein, setzen Sie den Cursor auf den Namen und rufen Sie dann in RStudio mit der Taste F1 die Hilfe auf. Alternativ gelangen Sie auch zur Hilfe, indem Sie dem Funktionsnamen vorangestellt ein `?` eintippen und die entsprechende Zeile im Skript oder in der Konsole ausführen, zum Beispiel

```
Call:
lm(formula = favorites ~ retweets, data = tweets)

Residuals:
    Min      1Q  Median      3Q     Max
-13.798  -5.893  -2.400   2.064  56.056

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   5.9727     2.9192   2.046   0.0477 *
retweets      1.9709     0.8987   2.193   0.0345 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 11.79 on 38 degrees of freedom
  (35 Beobachtungen als fehlend gelöscht)
Multiple R-squared:  0.1123, Adjusted R-squared:  0.08898
F-statistic: 4.809 on 1 and 38 DF,  p-value: 0.0345
```

Kennwerte zum gesamten Modell: R², F-Wert und fehlende Daten

Kennwerte zu den Regressionskoeffizienten: Gewicht (Estimate), Standardfehler (Std. Error) und Signifikanzwert (Pr).

Verteilung der Residuen: durch das Modell nicht erklärte Streuung

Abb. 5.7 Output einer Regressionsanalyse in R. Das Beispiel dient nur zu Demonstrationszwecken, es handelt sich um fiktive Daten. (Quelle: eigene Darstellung)

Tab. 5.6 R-Basisfunktionen für die Datenanalyse

Funktion	Beschreibung
`table()`	Häufigkeiten und Kreuztabellen
`mean()`	Mittelwerte
`sd()`	Standardabweichungen
`summary()`	Fünf-Punkte-Zusammenfassung (für metrische Variablen)
`cor.test()`	Korrelation inklusive Signifikanztest
`lm()`	Lineare Modelle (Regression)
`plot()`	Grafiken, beispielsweise Streudiagramme
`boxplot()`	Boxplots

Quelle: eigene Darstellung

`?table`. Eine weiterführende Übersicht über gängige Szenarien der statistischen Datenanalyse und wie sie mit R umgesetzt werden, finden Sie auf der Quick-R-Seite bei Kabacoff[16] oder in Field, Miles & Field (2012).

5.1.5 Grafiken

Ein zentraler Teil der Datenauswertung besteht in der Visualisierung von Ergebnissen. Vor der Gestaltung von Grafiken sollte man sich zunächst vor Augen halten, welche Daten visualisiert und welche Erkenntnisse dadurch gewonnen werden sollen. Wichtig ist bei der Interpretation immer, dass eine Grafik nur eine Repräsentation des Gegenstandes und nicht der Gegenstand selbst ist.

Inspiration und Lösungsvorschläge für die Visualisierung mit R finden Sie in der R Graph Gallery.[17] Das Paket *ggplot2* stellt nicht nur eine beeindruckende Menge an Funktionen bereit, sondern integriert die Funktionalitäten in ein aufgeräumtes Gesamtkonzept und etabliert damit eine Art Grafiksprache.[18] Die Grundidee besteht darin, dass Daten visuellen Elementen zugeordnet (engl. *mapping*) werden und diese anschließend durch das Übereinanderlegen von Schichten mit Gestaltungselementen grafisch aufbereitet werden. Die Schichten enthalten Informationen wie den Diagrammtyp, die Farben und Achsenbeschriftungen.

Grafiken erstellen Um eine Grafik zu erstellen, wird zunächst das Package *ggplot2* geladen und es wird die Funktion `ggplot()` aufgerufen. Dieser Funktion

[16] Siehe Kabacoff (2017; https://www.statmethods.net).

[17] Siehe Holtz (2018; https://www.r-graph-gallery.com/).

[18] Das Konzept geht zurück auf das Buch „The Grammar of Graphics“ (Wilkinson 1999).

werden Angaben mitgegeben, die für die gesamte Grafik gelten. Als erstes Argument nimmt sie den Datensatz, der visualisiert wird, entgegen. Als zweites Argument wird die Funktion `aes()` zur Zuordnung von grafischen Elementen zu Daten angegeben. Dabei wird unter anderem festgelegt, welche Daten auf die x- und welche auf die y-Achse geplottet werden oder anhand welcher Merkmale die Grafik eingefärbt werden soll. Zu diesem Grundobjekt werden anschließend durch das Plus-Zeichen + weitere Schichten hinzugefügt. So benötigt es mindestens eine weitere Schicht, um den Diagrammtyp festzulegen. Welche Diagrammtypen sich jeweils eignen, hängt von den Datentypen ab (Abb. 5.8):

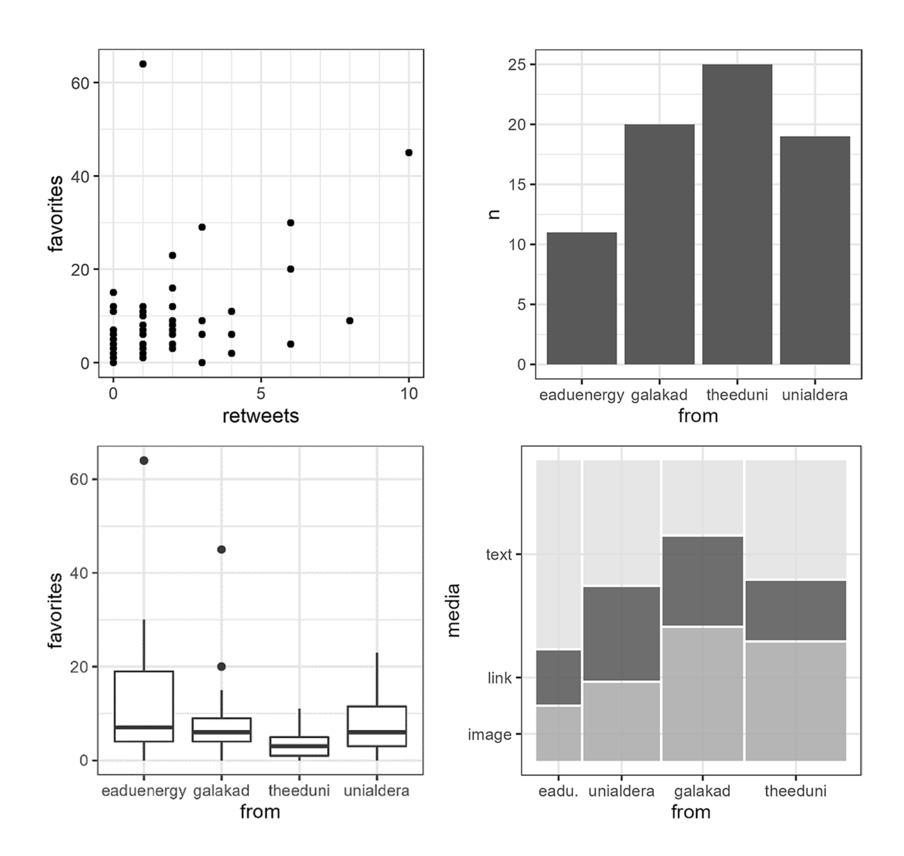


Abb. 5.8 Streudiagramm (links oben), Säulendiagramm (rechts oben), Boxplot (links unten) und Mosaikplot (rechts unten) mit *ggplot2*. Basis: 75 Fantasie-Tweets, nur zu Demonstrationszwecken geeignet. (Quelle: eigene Darstellung)

- **Streudiagramme** eignen sich besonders dann, wenn **zwei metrische Variablen** ins Verhältnis gesetzt werden. Ein Streudiagramm wird mit dem Funktionsaufruf `geom_point()` erstellt. Soll beispielsweise das Verhältnis zwischen den Variablen `favorites` und `retweets` ermittelt werden, können diese Daten direkt auf die x- und y-Achse projiziert werden:

```
ggplot(tweets, aes(x = retweets, y = favorites)) +
  geom_point()
```

- **Boxplots** visualisieren die Verteilung der Werte einer **einzelnen metrischen Variablen**, gegebenenfalls getrennt nach Gruppen. Die notwendigen Parameter für die Verteilung werden dabei von der Funktion `geom_boxplot()` selbst ermittelt. Um die Verteilung der Favorites je Account zu sehen, können die Namen der Accounts auf die x-Achse und die Favorites auf die y-Achse geplottet werden:

```
ggplot(tweets, aes(x = from, y = favorites)) +
  geom_boxplot()
```

- Liegen **kategoriale Daten** vor, kann deren Anzahl über **Balken- oder Säulendiagramme** dargestellt und verglichen werden. Die einzelnen Werte einer Kategorie werden zuvor ausgezählt. Anschließend wird das Balken- bzw. Säulendiagramm mit `geom_bar()` bzw. `geom_col()` erstellt. Um auszuzählen, wer am meisten Tweets verfasst hat, kann die jeweilige Anzahl der Tweets über `count(from)` ermittelt und in der Spalte `n` abgelegt werden. In der `aes()`-Funktion wird dann die Spalte `from` auf die x-Achse und der Wert `n` auf die y-Achse gemappt:

```
tweets %>%
  count(from) %>%
  ggplot(aes(x = from, y = n)) +
  geom_col()
```

- Die Kombination **mehrerer Kategorien** kann in **Mosaikplots** visualisiert werden. Ähnlich zu einem Balkendiagramm entspricht der Flächeninhalt (bestimmt durch Breite und Höhe) der Anzahl der Datensätze mit der jeweiligen Kombination. Hierzu eignet sich das Package *ggmosaic* (Jeppson et al. 2021). Die Kombination wird in diesem Fall direkt in `geom_mosaic()` über die `product()`-Funktion definiert:

```
library(ggmosaic)
tweets %>%
  ggplot() +
  geom_mosaic(aes(product(media, from)))
```

Mit *ggplot* und entsprechenden Zusatzpaketen können viele weitere interessante Grafiken wie Heatmaps, Konturdiagramme, Karten mit geografischen Informationen oder Netzwerke erstellt werden. Weitere Anregungen und Hinweise auf Tutorials finden Sie unter anderem auf der Webseite des *ggplot*-Pakets oder im Buch „R for Data Science" (Wickham und Grolemund 2016).[19]

Grafiken gestalten Um Grafiken weiter auszugestalten, werden Parameter in den aufgerufenen Funktionen angegeben oder zusätzliche Gestaltungselemente in Schichten mit einem + angehängt.[20] Mögliche Optionen, um das Streudiagramm aus Abb. 5.8 weiter zu gestalten, sind:

- **Farbe**: Möchte man Farbe ins Spiel bringen, wird in der `aes()`-Funktion festgelegt, anhand welcher Variable die Grafik eingefärbt werden soll, etwa um für jeden Account eine eigene Farbe zu verwenden. Standardmäßig ist dabei eine Farbpalette von *ggplot2* hinterlegt, die bei Bedarf aber auch abgeändert werden kann – etwa mittels einer zusätzlichen Schicht:

```
scale_fill_manual(
  values = c("blue", "green", "red", "yellow")
)
```

  Dabei müssen so viele Farben angegeben werden, wie es Ausprägungen der eingefärbten Variable gibt.
- **Positionierung**: Die Gestaltung der Punkte des Streudiagramms kann verändert werden, indem der `geom_point()`-Funktion weitere Parameter mitgegeben werden. Über `position = "jitter"`, werden die Punkte je ein klein wenig von ihrer eigentlichen Position verschoben, wodurch überlappende Werte sichtbar werden. Alternativ könnte man auch einen

[19] Siehe Wickham et al. (2022d; https://ggplot2.tidyverse.org/#learning-ggplot2) und Wickham und Grolemund (2016; https://r4ds.had.co.nz/data-visualisation.html).

[20] Eine umfängliche Auflistung aller Befehle zum Gestalten von Plots mithilfe von *ggplot2* findet sich in der Dokumentation bei Wickham et al. (2022c; https://ggplot2.tidyverse.org/reference/).

alpha-Wert setzen, der die Punkte transparenter macht. Welche Optionen in einem Diagramm möglich sind, ist in der Hilfe zum jeweiligen geom-Objekt dokumentiert. Säulen können beispielsweise gestapelt werden, indem die Funktion geom_col() um den Parameter position = "stack" ergänzt wird.

- **Skalen**: Liegen schiefe Verteilungen vor, dann werden die Fälle mit kleineren Werten besser sichtbar, wenn man die Skala transformiert. Für eine Umwandlung der y-Achse in eine logarithmierte Skala wird die Schicht scale_y_log10() hinzugefügt, analog dazu wird die x-Achse mit der zusätzlichen Schicht scale_x_log10() transformiert. 0-Werte werden durch das Logarithmieren unendlich groß und sind damit nicht darstellbar. Ein einfacher Trick besteht darin, alle Werte vor dem Logarithmieren um eins zu erhöhen, beispielsweise durch x = favorites + 1. Dadurch gehen keine Werte verloren, solche Umwandlungen dürfen jedoch nicht bei der anschließenden Interpretation übersehen werden.[21]
- **Layout**: Mitunter ist es übersichtlicher, für jede Kategorie eines Datensatzes eine eigene Grafik zu erstellen, das heißt zu facettieren. Mit der Schicht facet_wrap(~ from) würde für jeden Account eine einzelne Teilgrafik erstellt werden.
- **Beschriftungen**: Achsenbeschriftungen und Titel lassen sich über die Funktionen labs() und ggtitle() hinzufügen.[22]
- **Themes**: Das allgemeine Erscheinungsbild von Grafiken lässt sich durch Themes steuern. Im folgenden Beispiel wird über themes_bw() ein einfaches Schwarz-Weiß-Thema verwendet und über den Parameter base_size wird die zugrunde liegende Schriftgröße aller Textelemente festgelegt. Tipp: Um ein solches Thema für alle Grafiken in einem Skript vorzugeben, wird nach dem Laden der Packages der Befehl theme_set(theme_bw()) verwendet. Einzelne Elemente, wie die Formatierung der Legenden, lassen sich anschließend über den themes()-Befehl justieren.

Der Code mit den zusätzlichen Parametern und Schichten zur Ausgestaltung des Streudiagramms (Abb. 5.9) könnte also wie folgt aussehen:

[21] Zur Darstellung schiefer Verteilungen siehe auch Newman 2005.

[22] Achten Sie im Rahmen Ihrer wissenschaftlichen Arbeit stets darauf, Ihre Grafiken mit Angaben zu versehen, die für die Einordnung und Interpretation notwendig sind, insbesondere die Anzahl der dargestellten Fälle (die Basis).

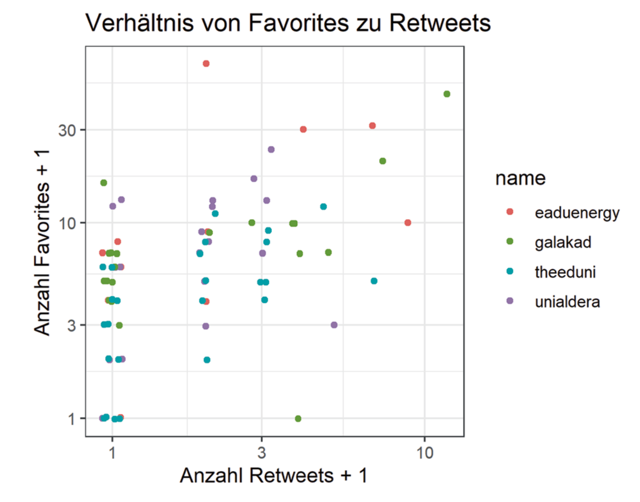


Abb. 5.9 Farbiges Streudiagramm mit *ggplot2*. Basis: 75 Fantasie-Tweets, nur zu Demonstrationszwecken geeignet. (Quelle: eigene Darstellung)

```
ggplot(tweets, aes(
    y = favorites + 1,
    x = retweets + 1,
    color = from)
  ) +

  # Position verschieben
  geom_point(position = "jitter") +

  # Logarithmieren
  scale_y_log10() +
  scale_x_log10() +

  # Beschriftungen hinzufügen
  labs(
    y = "Anzahl Favorites + 1",
    x = "Anzahl Retweets + 1"
  ) +
  ggtitle("Verhältnis von Favorites zu Retweets") +

  # Thema setzen
  theme_bw(base_size = 12)
```

In der Grafik lässt sich also die Tendenz erkennen, dass häufiger geteilte Tweets auch mehr Favorites erhalten, wobei einige Accounts wie unialdera stärker als andere von diesem Muster abweichen. Die Punkte am linken Rand deuten (unter Berücksichtigung der Logarithmierung) außerdem darauf hin, dass viele Tweets gar nicht retweetet wurden – und dennoch mal mehr und mal weniger Favorites erhalten.

Wenn Sie Grafiken erstellen, erhalten Sie wahrscheinlich früher oder später Warnmeldungen. Die Meldung „removed 35 rows containing missing values (geom_point)“ ist etwa ein Hinweis darauf, dass der Datensatz fehlende Werte (`NA`) beinhaltet. Je nachdem, wofür ein `NA` in einer Zeile steht, können diese Werte vorher unterschiedlich behandelt werden. Mittels `replace_na()` können sie durch `0` ersetzt oder durch Filtern mit der Bedingung `!is.na()` entfernt werden. Auch wenn 0-Werte logarithmiert werden, erscheint eine Warnmeldung wie „Transformation introduced infinite values in continuous y-axis.” Wenngleich nicht jede Warnung problematisch sein muss, sollten Sie diese bei der Interpretation der Grafiken berücksichtigen.

Grafiken speichern Grafiken können als Bilddatei abgespeichert werden, zum Beispiel im PNG-Format. Das Abspeichern erfolgt über die Funktion `ggsave()` – wobei standardmäßig immer die letzte Grafik im PLOTS-Reiter abgespeichert wird. Diese Funktion benötigt als Parameter zunächst den gewünschten Dateinamen, optional können Hinweise zur Auflösung und zum Format mitgegeben werden, etwa über die Parameter `dpi` (= dots per inch), `width` oder `height`:

```
ggsave(
  "streudiagram.png",
  dpi = 300,
  width = 10,
  height = 10, unit = "cm"
)
```

Für manuelle Nacharbeiten empfiehlt sich alternativ das SVG-Format, das mit kostenlosen Programmen wie Inkscape[23] weiterbearbeitet werden kann.

[23] Inkscape ist ein Programm zur Bearbeitung von Vektorgrafiken (Inkscape 2022; https://inkscape.org/).

Übungsfragen

1. Warum sollten Sie in R immer in einem Projekt arbeiten?
2. Inwiefern unterscheiden sich die Arbeit mit einem Skript und das Arbeiten in der Konsole?
3. Wie lauten die Tastenkombinationen für den Zuweisungsoperator und für die Pipe?
4. Was ist der Unterschied zwischen einer For-Schleife und einer vektorisierten Funktion?
5. Wie können Sie Excel-Dateien in R einlesen?
6. Was bedeutet der Ausdruck `NA` in R?
7. Was ist ein Tibble und was ist das Tidyverse?
8. Mit welchen Basis-R-Funktionen und mit welchen Funktionen aus dem Tidyverse können Sie Zeilen und Spalten auswählen?
9. Wie zählen Sie in R in einem Datensatz die Anzahl der Fälle je Kategorie aus?
10. Mit welchen Diagrammtypen können Sie anschaulich metrische Daten darstellen, mit welchen kategoriale Daten?

Weiterführende Literatur

Durch die starke Community findet sich eine Vielzahl sehr guter Hilfestellungen rund um R im Web. Die zentrale Anlaufstelle ist The Comprehensive R Archive Network (CRAN 2022; https://cran.r-project.org). Dort werden auch die meisten Packages verwaltet. Die Dokumentation der Packages kann nicht nur über die Hilfefunktion in RStudio, sondern etwa in der Form von PDF-Dateien auch über CRAN bezogen werden. Eine Einführung in die Basisfunktionen von R findet sich in:

Philips, N. D. (2018). *YaRrr! The Pirate's Guide to R.* https://bookdown.org/ndphillips/YaRrr/.

Eine umfassende Einführung in die Datenanalyse mit einem Fokus auf das Tidyverse finden Sie in:

Wickham, H. & Grolemund, G. (2017). *R for Data Science.* Sebastopol: O'Reilly. http://r4ds.had.co.nz.

Die genannten Bücher stehen unter einer Creative Commons-Lizenz, sind sowohl kostenlos online verfügbar als auch in gedruckter Form im Buchhandel käuflich.

Typische Lösungen für typische Probleme, aber auch eine übersichtliche Einführung, finden Sie bei Kabacoff (2017; https://www.statmethods.net). Spezielle Probleme werden häufig bei Stack Overflow (2022; https://stackoverflow.com) besprochen, hier können Sie auch selbst Fragen stellen. Gerade in der ersten Lernphase einer Sprache sind Cheat Sheets hilfreich, auf denen die wichtigsten Funktionen knapp zusammengefasst sind. Eine Zusammenstellung solcher Blätter bietet die Webseite von RStudio (2022b; https://www.rstudio.com/resources/cheatsheets/). Darüber hinaus gibt es einige spezialisierte Bücher für die Analyse mit R, in den folgenden Kapiteln finden Sie dazu weitere Hinweise.

5.2 Einführung in die Datenanalyse mit Python

Python ist eine vergleichsweise leicht zu erlernende Programmiersprache. Die Einsatzgebiete sind dennoch sehr breit. Python-Programme findet man zum Beispiel bei der Navigation von Flugzeugen, der Beleuchtung von Schiffen oder der Erstellung von Webseiten. Einige Spezialeffekte in Filmen wie Star Wars basieren auf Python-Programmen (Python Software Foundation 2019a). Auch die Namensgeber kommen aus dem Filmbereich: Die Benennung geht auf das Komikerensemble Monty Python zurück (Python Software Foundation 2019b).

In der wissenschaftlichen Datenanalyse sind die Einsatzgebiete ebenfalls breit gefächert. Python wird für die Datenerhebung zum Beispiel beim Webscraping genauso verwendet wie für die automatisierte Bilderkennung mit künstlichen neuronalen Netzwerken (KNN). Für viele Aufgaben stehen fertige Programmierbibliotheken zur Verfügung, sodass der Entwicklungsaufwand selbst bei komplexen Verfahren überschaubar ist. Bei der Entwicklung von Python-Skripten kommen in der Regel drei Komponenten zusammen:

- Der **Python**-Interpreter, der die Python-Quelltexte abarbeitet, die Skripte also ausführt (Python Software Foundation 2022c).
- **Programmbibliotheken**, die zusätzliche Funktionen bereitstellen, zum Beispiel *pandas* (Pandas development team 2022a) und *numpy* (Harris et al. 2020) für die Datenanalyse. Der Paketmanager pip hilft bei der Installation von zusätzlichen Programmbibliotheken (The pip developers 2022).

- Eine **Entwicklungsumgebung** wie JupyterLab (Jupyter 2022), Spyder (2022) oder PyCharm (Jetbrains 2022a). Diese sind sehr hilfreich, aber nicht zwingend notwendig. Skripte können auch mit einem einfachen Texteditor wie Atom (GitHub 2022a) oder Notepad++ (Ho 2022) entwickelt werden.

Die folgenden Abschnitte führen nach und nach in diese Komponenten und erste Programmbefehle von Python ein. Es ist eher unwahrscheinlich, dass alles auf Anhieb gelingt. Die Fehlersuche gehört zum Programmieren dazu und ist eine wesentliche Quelle der Erkenntnis. Oft hilft es, Fehlermeldungen wörtlich in eine Suchmaschine einzugeben. Hilfe finden Sie außerdem in Foren wie Stack Overflow.[24] Diese Einführung wird kaum dafür ausreichen, dass Sie anschließend selbstständig Skripte entwickeln können. Das ist zu Beginn aber auch nicht nötig. Viel wichtiger ist, dass Sie vorhandene Skripte, die Sie zum Beispiel in Foren oder den anderen Kapiteln des Buchs finden, nachvollziehen und anpassen können. Suchen Sie sich anschließend ein eigenes Projekt und versuchen Sie sich an der Umsetzung!

5.2.1 Entwicklungsumgebung JupyterLab

Als Entwicklungsumgebung für den Einstieg in Python bietet sich JupyterLab an. Das zentrale Konzept dieser Entwicklungsumgebung sind Notebooks. Ein Notebook ist eine Datei, in der sowohl der Programmcode als auch die Ergebnisse zusammengefasst sind. Zusätzlich lassen sich mit Markdown[25] erklärende Texte verfassen (Abb. 5.10). Das ist eine gute Kombination für die Datenanalyse, weil gleichzeitig die Ergebnisse und die einzelnen Schritte dokumentiert werden. JupyterLab enthält noch einige weitere nützliche Funktionen, insbesondere können Terminals (= Kommandozeilen) oder auch R-Skripte ausgeführt werden. Außerdem ist der Weg zum Cloud Computing ausgehend von Jupyter Notebooks nicht mehr weit, falls Sie an die Grenzen der Rechenkapazitäten Ihres Computers stoßen. Beim Cloud Computing werden die Python Skripte nicht auf dem eigenen Computer ausgeführt, sondern auf der Server-Infrastruktur von Cloud-Dienstleistern (siehe Abschn. 6.3).

[24] Siehe Stack Overflow (2022; https://stackoverflow.com).

[25] Markdown ist eine Sprache zur Formatierung von Texten (siehe Abschn. 3.2). Überschriften werden zum Beispiel mit Rauten und Hervorhebungen mit Sternchen erzeugt.

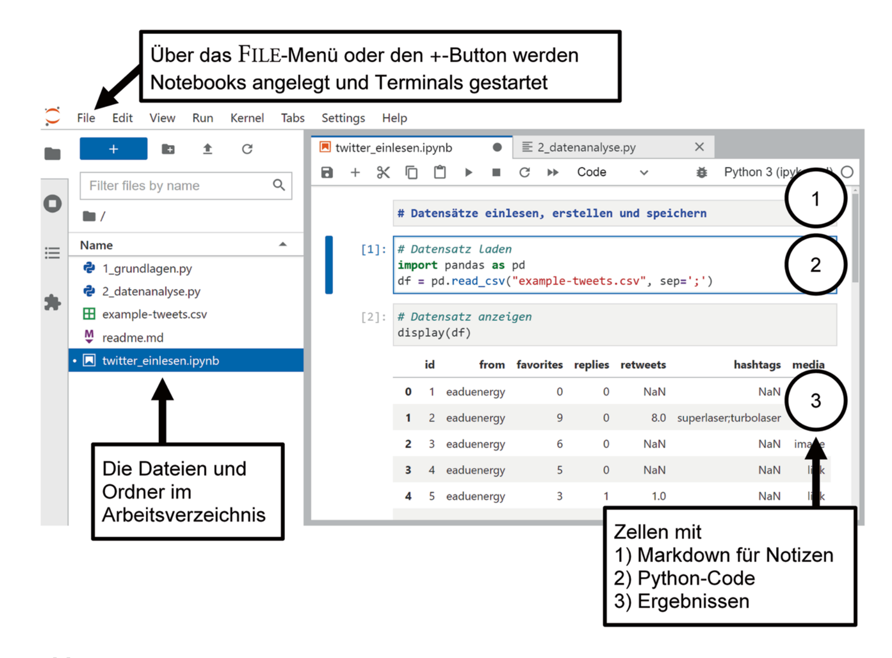


Abb. 5.10 Die Oberfläche von JupyterLab. (Quelle: eigene Darstellung)

Python installieren Für die Installation von Python gibt es mehrere alternative Wege. Sie müssen nicht zwangsläufig jede der Komponenten – Python, Packages, Entwicklungsumgebung – einzeln installieren, sondern können auf sogenannte Distributionen zurückgreifen, die für das jeweilige Betriebssystem vorkonfektioniert sind. Eine für Windows, macOS und Linux zugeschnittene Distribution ist Anaconda (2020). Alternativ bietet sich für Windows-Nutzer auch WinPython (2022) an.

Beide Distributionen stellen Python in einem abgegrenzten Bereich zur Verfügung, sodass gleichzeitig auch mehrere Python-Versionen oder mehrere Distributionen auf dem gleichen Computer genutzt werden können. Wenn in mehreren Projekten unterschiedliche Python- oder Package-Versionen verwendet werden sollen, lassen sich Projekte zudem voneinander über sogenannte virtuelle Umgebungen (engl. *virtual environment*, *venv*) abkapseln – ein Thema, das für Anfänger meist noch nicht von Bedeutung ist. Eine Einführung finden Sie in der Python-Dokumentation.[26]

[26] Siehe Python Software Foundation (2022d; https://docs.python.org/3/tutorial/venv.html).

Ein Nachteil solcher Zusammenstellungen ist jedoch, dass damit eine weitere Komponente ins Spiel kommt und somit eine weitere mögliche Fehlerquelle. Es mag kontraintuitiv erscheinen, gerade für den Einstieg kann aber die im Folgenden beschriebene Installation der einzelnen Komponenten dabei helfen, die verschiedenen Teile und das Zusammenspiel nach und nach besser zu verstehen.

In jedem Fall benötigen Sie Python. Laden Sie eine aktuelle Version von https://www.python.org/downloads/ herunter. Es muss nicht die allerneueste Version sein, ältere Versionen sind in der Regel praxiserprobter. Wenn Sie bei der Installation auf Schwierigkeiten stoßen, versuchen Sie es mit einer anderen Version. Auf Mac-Computern und unter Linux besteht eine hohe Wahrscheinlichkeit, dass Python schon installiert ist. Trotzdem ist es ratsam, nicht mit dieser Systemversion zu arbeiten und eine aktuelle Version zu installieren. Achten Sie unter Windows bei der Installation darauf, die Option ADD PYTHON TO PATH zu aktivieren. Nur dann ist Python von jedem Ordner aus über die Kommandozeile nutzbar.[27] Die Erstinstallation kann etwas umständlich sein, durch diese Prozedur müssen Sie aber nur einmalig durchgehen. Wenn Sie Probleme mit der Installation haben, finden Sie online viele weitere Hinweise.[28]

Erstellen Sie nach der Installation ein leeres Arbeitsverzeichnis auf Ihrem Computer und öffnen Sie dort eine Kommandozeile (siehe Abschn. 1.2.2). Geben Sie nun auf der Kommandozeile `python` ein (Abb. 5.11). Wenn die Installation geklappt hat, wird die Python-Version angezeigt und innerhalb der Kommandozeile wird eine neue Kommandozeile gestartet, die Python-Befehle ausführt.[29] Sie können das Terminal testweise wie einen Taschenrechner benutzen, geben Sie einmal eine Rechenaufgabe wie `4096 / 23` ein und bestätigen Sie mit der Entertaste. Verlassen Sie die Python-Kommandozeile schließlich wieder mit dem Befehl `quit()`, damit bleibt das Fenster geöffnet, aber Sie befinden sich anschließend wieder auf der Ausgangsebene.

---

[27] Falls Sie diese Option verpassen, können Sie den Pfad selbst zur PATH-Variable hinzufügen. Dazu müssen Sie den Ordner kennen, in dem Python installiert wurde. Diesen Ordner legen Sie bei der Installation fest, alternativ können Sie auf Ihrem Computer nach „python.exe“ suchen. Fügen Sie den gefundenen Pfad (ohne python.exe) sowie den Unterordner *Scripts* zu den Umgebungsvariablen hinzu, über die Kommandozeile: `SETX PATH "%PATH%; C:\Users\Nutzername\Python3; C:\Users\Nutzername\Python3\Scripts"`. Die beiden hier angegebenen Pfade müssen Sie an Ihr System anpassen. Damit die Änderungen aktiv werden, schließen Sie anschließend die Kommandozeile und öffnen sie neu.

[28] Zum Beispiel bei Reitz (2021; https://docs.python-guide.org).

[29] Falls das nicht funktioniert, können Sie nach der Installation von Python 3 die Befehle `python3` und später `pip3` verwenden.

Eingabeaufforderung

```
C:\Users\Jakob>python
Python 3.7.2 (tags/v3.7.2:9a3ffc0492, Dec 23 2018, 23:09:28
) [MSC v.1916 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more i
nformation.
>>> quit()

C:\Users\Jakob>
```

Abb. 5.11 Python auf der Windows-Kommandozeile. (Quelle: eigene Darstellung)

JupyterLab einrichten Für die Entwicklung von Skripten arbeiten Sie besser nicht direkt mit der Kommandozeile, sondern mit einer Entwicklungsumgebung wie JupyterLab. JupyterLab und alle weiteren Komponenten lassen sich aber gut von der Kommandozeile aus installieren:

- Aktualisieren Sie zunächst den Paketmanager pip auf die neueste Version:

  ```
  python -m pip install --upgrade pip
  ```

- Installieren Sie dann Jupyter und JubyterLab, beides benötigt eine Weile:

  ```
  pip install jupyter
  pip install jupyterlab
  ```

- Installieren Sie gleich noch die folgenden häufig benötigten Packages:

  ```
  pip install numpy pandas scipy matplotlib
  ```

Weitere Pakete können auch später noch mit dem Paketmanager pip installiert werden.

- Packages werden mit der Zeit weiterentwickelt, sodass neue Funktionen dazukommen oder alte abgeschafft werden. Welche Versionen Ihre Packages haben, können Sie über den Befehl `pip list` abfragen. Ist eine neuere Packageversion verfügbar und wollen Sie diese updaten, verwenden Sie den `upgrade`-Parameter:

```
pip install --upgrade numpy
```

- Starten Sie nun JupyterLab von der Kommandozeile. Achten Sie darauf, dass die Kommandozeile im Arbeitsverzeichnis (siehe Kap. 1) geöffnet ist:

```
jupyter lab
```

Es sollte sich der Browser öffnen und die Oberfläche von JupyterLab anzeigen (Abb. 5.10). Zum Beenden können Sie später das Browserfenster und die Konsole einfach wieder schließen.

Nach dem Start ist eine Übersicht über die vorhandenen Dateien im Arbeitsverzeichnis, von dem aus JupyterLab gestartet wurde, zu sehen. Legen Sie über den Menüpunkt FILE ein neues Python-3-Notebook an. Geben Sie dem neuen Dokument einen Namen, indem Sie es beispielsweise über das Menü (FILE > RENAME NOTEBOOK) umbenennen. Das Notebook besteht aus einzelnen Zellen, die Programmcode oder Texte im Markdown-Format (siehe Abschn. 3.2) enthalten. Ganz oben finden Sie eine leere Code-Zelle. Wenn Sie in eine Code-Zelle etwas eingeben und über das RUN-Menü oder den entsprechenden Button in der Toolbar (Abb. 5.12) den Code ausführen, dann wird darunter die Ausgabe dargestellt. Zum Ausführen einer Zelle können Sie auch die Tastenkombination Strg + Enter (Windows) bzw. Command + Enter (Mac) verwenden. Über Shift + Enter wird ebenfalls eine Zelle ausgeführt und gleichzeitig springt der Fokus in die nächste Zelle – so können Sie sich schrittweise durch das vollständige Skript bewegen.

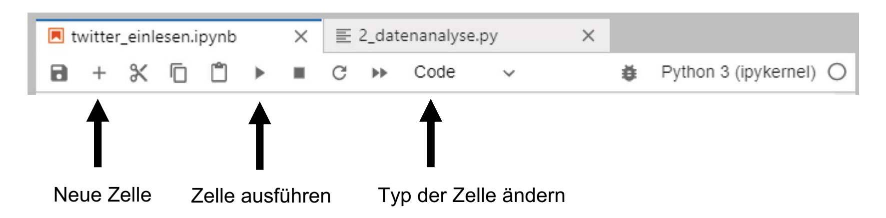


Abb. 5.12 Die Toolbar in Jupyter Notebooks. (Quelle: eigene Darstellung)

5.2.2 Die wichtigsten Programmbefehle im Schnelldurchlauf

Werte und Listen Um sich mit Python vertraut zu machen, versuchen Sie die folgenden Beispiele in JupyterLab nachzuvollziehen. Geben Sie dafür Schritt für Schritt die Befehle in eine neue Zelle ein und führen Sie diese Zelle aus. Immer wichtig ist es, den eigenen Code zu kommentieren. Alle Angaben hinter einer Raute gelten als Kommentar und werden nicht ausgeführt:

```
# Ich bin ein Kommentar
```

Ein Skript besteht außerdem aus Befehlen, die nacheinander abgearbeitet werden. Ein Befehl hat einen Namen und nimmt Parameter[30] entgegen, die in Klammern angegeben werden. Der `print()`-Befehl gibt zum Beispiel Text aus, wobei Text immer in einfache oder doppelte Anführungszeichen gesetzt wird:

```
print("Hello world!")
```

Das letzte Objekt in einer Zelle wird auch ohne `print()`-Befehl ausgegeben. Testen Sie eine einfache Rechenaufgabe:

```
23 + 1
```

Werte können in Objekten abgespeichert werden, um sie anschließend in weiteren Befehlen nutzen zu können. Diese Objekte werden mit dem Ist-gleich-Zeichen = definiert, wobei der Name des Objektes davor und der Wert des Objektes dahinter steht:

```
meinname = "Eliza"
```

Mit dem Pluszeichen + können Zeichenketten verbunden werden. Beachten Sie im folgenden Beispiel, dass einerseits eine Zeichenkette in Anführungszeichen und andererseits das soeben definierte Objekt verwendet werden. Das Objekt ist hier ein Platzhalter für den vorher abgelegten Wert:

```
print("Mein Name ist " + meinname)
```

[30] Parameter werden auch Argumente genannt.

Eine Liste wird in eckigen Klammern angegeben, wobei die Elemente mit Kommata getrennt werden:

```
eigenschaften = ["schön", "reich", "intelligent"]
```

Auf die Elemente von Listen kann mit Indizes zugegriffen werden. Indizes geben an, an der wievielten Stelle in der Liste sich das gesuchte Element befindet. Dafür wird die Nummer eines Elements in eckigen Klammern angegeben. Die Zählung beginnt – ein Unterschied zu R – in Python mit 0:

```
print(eigenschaften[0])
```

Bislang haben Sie Zeichenketten aneinandergehängt und Zahlen addiert. Weitere Funktionen können Sie aus Programmbibliotheken nachladen. Dazu dient der `import`-Befehl. Mit der Bibliothek *re* können Sie zum Beispiel reguläre Ausdrücke verwenden (siehe Abschn. 4.1.1).[31] Der Befehl `re.sub()` ersetzt ein Suchmuster (erster Parameter, hier alle Vokale) durch etwas anderes (zweiter Parameter, hier ein Unterstrich) innerhalb einer Zeichenkette (dritter Parameter, hier die Zeichenkette „schön“):

```
import re
ohnevokale = re.sub("[aoueiöäü]", "_", "schön")
```

Benutzen Sie den `print()`-Befehl, um sich das Ergebnis anzuschauen.

Bei der Arbeit mit Python spielen Leerzeichen eine besondere Rolle. Die Struktur des Quelltextes wird durch Einrückungen mit immer vier Leerzeichen gekennzeichnet. Mitunter drohen Zeilen dadurch sehr lang zu werden – man kann Code allerdings nicht beliebig auf neue Zeilen umbrechen, sondern muss sich an Konventionen halten. Um Code dennoch auf einer neuen Zeile fortsetzen zu können, wird der Backslash \ an das Ende einer vorangegangenen Zeile gesetzt – wir machen davon bei einigen der folgenden Beispiele Gebrauch.

Kontrollstrukturen und Funktionen Mit Kontrollstrukturen sind Sprachelemente gemeint, die den Ablauf eines Programms steuern. Um Befehle mehrfach auszuführen, werden Schleifen (engl. *loops*) eingesetzt. Eine besonders praktische

[31] In der Regel setzt man alle import-Anweisungen an den Anfang eines Skripts, sodass die benötigten Bibliotheken (= Abhängigkeiten) schnell erkennbar sind.

Schleife ist in Python die For-in-Schleife, die den folgenden Codeblock für jedes Element einer Liste einmal durchläuft. Die Syntax lautet:

```
for <element> in <liste>:
    # Weiterer Code, der mit
    # <element> arbeitet
```

Wichtig ist ein Doppelpunkt am Ende und dass die nächsten Zeilen eingerückt werden. Durch die Einrückung mit vier Leerzeichen werden zusammenhängende Codeblöcke erstellt. Das kann in JupyterLab mit der Tabulatortaste erreicht werden; der Editor ersetzt den Tabulator automatisch durch vier Leerzeichen:

```
for x in eigenschaften:
    satz = meinname + " ist " + x
    print(satz)
```

Es steht dabei frei, wie das Element benannt wird. Statt „x“ könnte auch „eigenschaft“ oder „item“ eingesetzt werden. Unter diesem Namen ist das Element innerhalb des eingerückten Codeblocks verfügbar. Der Name ist also beliebig, muss dann aber beibehalten werden.

Eine weitere Möglichkeit, um Funktionen auf Listen anzuwenden, sind List Comprehensions. Während die Verwendung einer For-in-Schleife ganz im Sinne imperativer Programmierung die einzelnen Schritte auflistet, sind List Comprehensions eine Variante funktionaler Programmierung:

```
saetze = ["Eliza ist " + x for x in eigenschaften]
```

In diesem Beispiel wird ausgehend von der Liste `eigenschaften` eine neue Liste `saetze` erstellt. Vor jedes Element `x` (der Name ist wiederum frei gewählt) der ursprünglichen Liste wird die Zeichenkette "Eliza ist " gestellt. Der Befehl wird ebenfalls für alle Elemente aus der Liste `eigenschaften` nacheinander ausgeführt, eine solche Schreibweise ist aber kürzer als eine Schleife. Selbst wenn dieses Muster zu Beginn schwer zu verstehen ist, nach etwas Übung wird die Entwicklung durch die kompakte Schreibweise sehr erleichtert.

Sie können das Ergebnis über den `print()`-Befehl ausgeben oder das Objekt `saetze` einfach in die letzte Zeile der Zelle setzen und die Zelle ausführen. Das letzte innerhalb einer Zelle erzeugte Objekt wird immer als Ergebnis ausgegeben.

Eine weitere wichtige Kontrollstruktur sind Bedingungen. Mit der If-Bedingung wird der Ablauf verzweigt, sodass ein Teil des Skripts nur unter der angegebenen Bedingung ausgeführt wird. Das lässt sich mit Schleifen kombinieren:

```
for x in eigenschaften:
    if x != "schön":
        print("Eliza ist " + x)
```

Mit dem Operator != werden die nach dem Doppelpunkt angegebenen, eingerückten Teile nur ausgeführt, wenn linke und rechte Seite der Bedingung *nicht* übereinstimmen. Sollen sie übereinstimmen, wird der Gleichheitsoperator == verwendet. Das Muster kann durch weitere Elif-Bedingungen ergänzt werden, wobei die späteren Blöcke nur ausgeführt werden, wenn nicht vorher schon eine Bedingung erfüllt war. Ein Else-Block am Ende wird immer ausgeführt, wenn unterwegs nichts anderes zugetroffen hat:

```
for x in eigenschaften:
    if x == "schön":
        print("Eliza war " + x)
    elif x == "reich":
        print("Eliza wird " + x + " sein")
    else:
        print("Eliza ist " + x)
```

Immer wenn Code mehrfach verwendet wird, sollte man eigene Funktionen definieren, damit die Skripte kürzer und übersichtlicher werden. Außerdem lassen sich Funktionen bei Bedarf leichter anpassen und austauschen, als wenn viele Stellen in einem Skript einzeln geändert werden müssten. Dieses Prinzip wird dry (= don't repeat yourself) genannt. In Python definieren Sie Funktionen mit dem Schlüsselwort `def`, gefolgt vom Namen der Funktion. Mögliche Parameter, die in der Funktion verarbeitet werden, sind in runden Klammern angegeben:

```
def superduper(eigenschaft):
    x = "sehr " + eigenschaft
    return (x)
```

Wie schon bei Schleifen kann der Parameter `eigenschaft` beliebig benannt werden, das Objekt steht innerhalb der Funktion unter diesem Namen zur Verfügung. Der Inhalt der Funktion, das heißt die einzelnen Befehle, werden wieder eingerückt. Am Ende der Funktion wird das Ergebnis über das Schlüsselwort `return` zurückgegeben. Eine Funktion nimmt also Parameter entgegen (Input), verarbeitet sie in einzelnen Schritten (Throughput) und gibt das Ergebnis zurück (Output). Diese Funktion kann anschließend beispielsweise innerhalb einer Schleife eingesetzt werden:

```
    for y in eigenschaften:
        print("Eliza ist " + superduper(y))
```

Beachten Sie hier zwei Dinge. Erstens können Funktionsaufrufe verschachtelt werden. Der `print()`-Aufruf nimmt den zusammengesetzten Wert `"Eliza ist " + superduper(y)` entgegen und darin wird die Funktion mit dem Namen `superduper()` aufgerufen. Zweitens müssen die Namen der übergebenen Objekte (hier `y`) nicht mit den Namen der Parameter aus der zuvor definierten Funktion (hier: `eigenschaft`) übereinstimmen. Es handelt sich im Beispiel bei `y` und `eigenschaft` um Platzhalter, die immer nur innerhalb der Funktion oder der Schleife gültig sind, das heißt, innerhalb der Funktion oder der Schleife werden dann alle diese Platzhalter durch die übergebenen Werte ersetzt. Die Parameternamen können aber beim Funktionsaufruf explizit angegeben werden, um für Klarheit zu sorgen:

```
    for x in eigenschaften:
        print("Eliza ist " + superduper(eigenschaft=x))
```

Wenn Skripte komplexer werden, helfen Funktionen dabei, die Übersicht zu behalten. Gleichzeitig nimmt bei umfangreichen Skripten die Wahrscheinlichkeit für Fehler zu. Deshalb ist es wichtig, Fehler von vornherein abzufangen. Hierzu dienen Try-except-Blöcke. Tritt innerhalb des Try-Blocks ein Fehler auf, dann wird direkt nach dem Fehler der Except-Block ausgeführt. Ein Finally-Block wird dagegen immer ausgeführt, egal ob ein Fehler aufgetreten ist oder nicht:

```
    eigenschaften = ["schön","reich",0,"intelligent"]

    for x in eigenschaften:
        try:
            x = "Eliza ist " + x
        except:
            x = "Eliza ist irgendwie anders"
        finally:
            print(x)
```

Das Beispiel wirft (engl. *raise*) deshalb einen Fehler aus, weil in einem Durchlauf der Schleife die Zeichenkette „Eliza ist “ mit der Zahl 0 addiert werden soll. Eine Addition von Zeichen und Zahlen kann vom Programm nicht sinnvoll interpretiert werden, da hier die Datentypen vermischt werden. Deshalb entsteht ein TypeError. Ein solcher Fehler entsteht immer, wenn der eingegebene Datentyp (in

dem Beispiel eine Zahl) nicht dem erwarteten Datentyp (im Beispiel eine Zeichenkette) entspricht. Es könnte aber versucht werden, genau diese TypeError-Fehler abzufangen (engl. *catch*) und die Zahl mit der `str()`-Funktion in eine Zeichenkette umzuwandeln:

```
for x in eigenschaften:
    try:
        x = "Eliza ist " + x
    except TypeError:
        x = "Eliza ist eine " + str(x)

    print(x)
```

Es ist durchaus ein guter Programmierstil, bewusst mit Fehlern zu arbeiten. Statt mit If-Kontrollstrukturen alle möglichen Bedingungen abzudecken, lässt man Programme gezielt in Fehler hineinlaufen, die dann behandelt werden.[32]

Dictionaries und Objekte Nicht nur die Skripte werden nach und nach komplexer, sondern auch die damit verarbeiteten Objekte. Neben einfachen Datentypen wie Zahlen oder Zeichenketten ist Ihnen oben bereits ein zusammengesetzter Datentyp begegnet, die Liste.[33] In Listen lassen sich mehrere Elemente ablegen, um sie nacheinander abzuarbeiten, etwa um mehrere Wörter miteinander zu verbinden. Welchen Sinn die einzelnen Wörter haben, geht aus einer Liste allerdings nicht hervor. Stellen Sie sich vor, eine Person soll durch unterschiedliche Eigenschaften beschrieben werden, das Alter, den Namen und den kognitiven Zustand. In Dictionaries (auch kurz Dicts genannt) können Sie diese verschiedenen Eigenschaften mit Name-Wert-Paaren erfassen. Dicts werden im Gegensatz zu Listen nicht mit eckigen, sondern mit geschweiften Klammern definiert:

```
ich = {'name': meinname, 'zustand': 'verwirrt'}
```

Beachten Sie, dass hier ein Wert wiederum als Objekt (`meinname`) angegeben werden kann. Dieses Objekt müsste vorher definiert oder durch eine Zeichenkette ersetzt werden. Auf die Werte der Elemente kann dann mit dem Namen zugegriffen werden:

[32] Weitere Ausführungen zu Fehlerbehandlungen finden Sie bei W3Schools (2022c; https://www.w3schools.com/python/python_try_except.asp).

[33] Siehe Abschn. 3.1 für eine Einführung in grundlegende Datentypen.

```
print(ich['name'] + " ist " + ich['zustand'])
```

Dicts enthalten lediglich Daten, das heißt Eigenschaften von den betrachteten Objekten. Bei der objektorientierten Programmierung kombiniert man dagegen Eigenschaften und Methoden in einem Objekt. Auch wenn es eine Weile braucht, um dieses Konzept zu begreifen, soll zumindest kurz auf die Grundlagen eingegangen werden.[34] In der objektorientierten Programmierung werden zunächst Klassen definiert, das sind eine Art selbstausgedachte Datentypen:

```
class Jedi:
    name = None
    staerke = 0

    def __init__(self, name):
        self.name = name
        self.staerke = 10

    def talk(self):
        print (self.name + " mein Name ist.")

        if self.staerke < 5:
            print("Mich schwach ich fühle.")
```

Das Beispiel erzeugt eine Jedi-Klasse mit den zwei Eigenschaften `name` und `staerke`, die mit leeren Werten vorbelegt werden. Hinzu kommen zwei Methoden – so nennt man die Funktionen einer Klasse oder eines Objekts. Die Methode `__init__` wird aufgerufen, sobald ein neues Objekt von dieser Klasse erzeugt wird, das heißt sobald die Klasse instanziiert wird. Die folgenden Beispiele erzeugen zwei voneinander unabhängige Instanzen und speichern sie in den Objekten `j1` und `j2` ab.

```
j1 = Jedi("Yoda")
j2 = Jedi("Rey")
```

Das Erzeugen von Instanzen aus einer Klasse sieht also ganz ähnlich aus wie ein Funktionsaufruf und es verhält sich auch so. Der Parameter hinter dem Klas-

[34] Eine allgemeine Einführung finden Sie bei W3Schools (2022b; https://www.w3schools.com/python/python_classes.asp).

sennamen wird direkt an die `__init__()`-Funktion übergeben und dort für die Einrichtung des Objekts verwendet.

Das konkrete aus der Klasse hervorgehende Objekt wird dann Instanz dieser Klasse genannt. Innerhalb der Methoden einer Klasse steht der Parameter `self` für genau diese eine konkrete Instanz der Klasse. Auf Eigenschaften und Methoden der Instanz kann zugegriffen werden, indem der Name mit einem Punkt angehängt wird, zum Beispiel innerhalb der Klassenmethoden mit `self.staerke`. Auf diese Weise können aber auch außerhalb der Klassendefinition Werte verändert werden:

```
j1.staerke = 2
```

Wenn Sie nun die Funktion der beiden Instanzen aufrufen, verhalten sich beide unterschiedlich, da sie einen unterschiedlichen Zustand haben – in der `talk()`-Methode wird `self` verwendet und der Zustand wurde vorher nur in einer der Instanzen verändert:

```
j1.talk()
j2.talk()
```

Die `talk()`-Funktion hat nur diesen einen Parameter `self`, der aber beim Aufrufen nicht angegeben wird, da hier automatisch immer die Instanz vorbelegt wird. Mit Klassen lassen sich genauso wie mit Funktionen komplexe Vorgänge abkapseln. Dabei muss man sich mit der inneren Funktionsweise einer Klasse nicht unbedingt auskennen, solange man weiß, wofür und wie man die Klasse verwendet.

Klassen erlauben es also, Dinge zu erzeugen, die Eigenschaften und Methoden aufweisen. Solche Dinge, die über Methoden verfügen, nennt man Objekte. Fast alle Dinge in Python sind Objekte und verfügen über Methoden. Das gilt zum Beispiel auch für Zeichenketten, wie sie in vielen der bisherigen Beispiele vorkamen. Zeichenketten verfügen über viele nützliche Methoden. Schlagen Sie in der Python-Dokumentation[35] nach und probieren Sie einige aus! Das folgende (vielleicht nicht ganz sinnvolle) Beispiel kettet solche Methodenaufrufe hintereinander:

```
"Yoda".lower().replace("y", "J").capitalize()
```

[35] Siehe Python Software Foundation (2022b; https://docs.python.org/3/library/stdtypes.html#text-sequence-type-str).

Zusammenfassend kann man sagen, dass in einem Skript vier verschiedene Zutaten verarbeitet werden:

- Einfache **Werte** wie Zahlen (`12`), Zeichenketten (`"Eliza"`) oder Wahrheitswerte (`True`, `False`) sowie zusammengesetzte Werte wie Listen, Dictionaries oder Objekte. Eigene Objekte mit komplexen Datentypen können über Klassen erzeugt werden.
- **Funktionen und Methoden**, mit denen Werte verarbeitet werden. Es lassen sich eigene Funktionen definieren oder Funktionen aus Bibliotheken importieren.
- **Kontrollstrukturen**, um den Ablauf eines Programms bzw. den Aufruf von Funktionen zu steuern. Dazu gehören For-in-Schleifen und If-else-Verzweigungen, zur Fehlerbehandlung können Try-except-Blöcke eingesetzt werden.
- **Kommentare** zur Dokumentation des Codes.

Diese typischen Zutaten werden immer wieder neu zusammengesetzt, wodurch das Programmieren nicht nur ein formaler, sondern auch ein sehr kreativer Prozess ist.

5.2.3 Datenanalyse mit Pandas

Neben den Standardfunktionen von Python kommen für die Datenanalyse in der Regel spezialisierte Programmbibliotheken zum Einsatz. Bibliotheken werden über den `import`-Befehl eingebunden. Sie enthalten Funktionen und Objekte, die die Arbeit deutlich erleichtern. Falls eine Bibliothek nicht auf dem System zur Verfügung steht, kann sie meistens über den Paketmanager pip nachinstalliert werden. Zu den wichtigsten Bibliotheken gehören:

- *numpy* (Harris et al. 2020) enthält Funktionen zum Umgang mit Zahlen und Matrizen. Das wichtigste Objekt in dieser Bibliothek sind mehrdimensionale Arrays vom Datentyp `ndarray`. Ein eindimensionales Array ist eine Liste, deren Elemente alle den gleichen Datentyp haben, meistens Zahlen. Erstellt man eine Liste, die andere Listen enthält (engl. *list of lists*), entsteht ein zweidimensionales Array. Zweidimensionale Arrays werden Matrizen (engl. *matrices*) genannt und die Dimensionen werden als Achsen (engl. *axes*) bezeichnet.
- *pandas* (Pandas development team 2022a) stellt den Typ `DataFrame` und dazugehörige Funktionen bereit, mit denen tabellenförmige Daten verarbeitet werden können. Die einzelnen Spalten weisen den Datentyp `Series` auf und wer-

den wie Listen behandelt. Die Datentypen von *pandas* sind mit den Datentypen von *numpy* kompatibel, enthalten aber weitergehende Funktionen für die Datenanalyse.
- *matplotlib* (Hunter et al. 2022) ist eine Bibliothek zum Erstellen von Grafiken.
- *sklearn* (= Scikit-learn; Pedregosa et al. 2011) wird für statistische Auswertungen verwendet, insbesondere für das maschinelle Lernen (siehe Abschn. 8.1). Hier sind Verfahren zur Regression, Klassifikation und zum Clustering implementiert, zum Beispiel für die Verwendung künstlicher neuronaler Netzwerke.

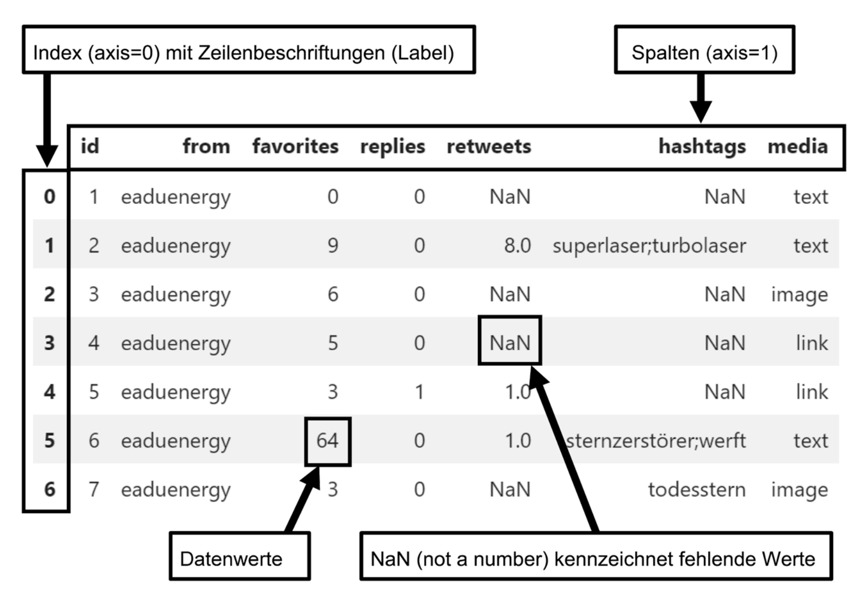
Jede dieser Bibliotheken eröffnet eine eigene kleine Welt mit vielen interessanten und nützlichen Ideen und Tools, aber auch eigenen Sprachen. Wenn Sie anfangen diese Sprachen zu lernen, schauen Sie ab und an in den Wörterbüchern – das heißt in den Dokumentationen – nach. Für die folgenden Schritte kann es sich lohnen, parallel die Dokumentationen von *pandas* und *numpy* zu öffnen, zu finden über eine Suchmaschine mit den Stichwörtern „reference“ oder auch „API reference“. Unten finden Sie weitere Hinweise, wie die Hilfe zu einzelnen Funktionen direkt aus JupyterLab geöffnet werden kann.

Datensätze einlesen und speichern Zu den Grundlagen der Datenanalyse gehört der Umgang mit den Dataframes der *pandas*-Bibliothek. Ein Dataframe folgt den Konventionen für Tabellen (Abschn. 4.2) und besteht aus nebeneinander angeordneten Spalten und untereinander angeordneten Zeilen (Abb. 5.13). An den Kreuzungen zwischen Spalten und Zeilen stehen die Werte. Nicht nur Spalten, sondern auch Zeilen können benannt sein. Normalerweise werden Zeilen einfach mithilfe von Nummern referenziert. Es können aber auch andere Namen (engl. *label*) vergeben werden. Diese Bezeichnungen heißen in *pandas* Index und sind für viele Operationen wichtig, etwa für das Sortieren oder das Zusammenfügen von Datensätzen.

Damit der Tippaufwand beim Benutzen von *pandas* reduziert wird und gleichzeitig erkennbar bleibt, aus welcher Bibliothek eine Funktion kommt, importiert man *pandas* häufig mit dem Alias `pd`:

```
import pandas as pd
```

Die *pandas*-Befehle sind anschließend immer am Präfix `pd` zu erkennen. Datensätze können darüber unter anderem im CSV-Format eingelesen werden. Im



	id	from	favorites	replies	retweets	hashtags	media
0	1	eaduenergy	0	0	NaN	NaN	text
1	2	eaduenergy	9	0	8.0	superlaser;turbolaser	text
2	3	eaduenergy	6	0	NaN	NaN	image
3	4	eaduenergy	5	0	NaN	NaN	link
4	5	eaduenergy	3	1	1.0	NaN	link
5	6	eaduenergy	64	0	1.0	sternzerstörer;werft	text
6	7	eaduenergy	3	0	NaN	todesstern	image

Abb. 5.13 Die Komponenten eines pandas-Dataframes. (Quelle: eigene Darstellung)

Beispiel wird der Datensatz *example-tweets.csv* eingelesen und die Daten werden im Objekt `df` abgelegt (☛ *Repositorium*):

```
df = pd.read_csv("example-tweets.csv", sep=";")
```

Weitere Optionen zum Anpassen der Funktion finden Sie in der Dokumentation der Packages. Sie können die Hilfe zu einem Befehl auch direkt aus dem Skript aufrufen, dazu wird an den Befehl ein Fragezeichen angehängt:

```
pd.read_csv?
```

Eine weitere Möglichkeit, die Optionen zu erkunden, bietet die Autovervollständigung in JupyterLab. Wenn Sie anfangen den Befehl einzutippen und dann nach dem Punkt die Tabulatortaste drücken, wird zunächst eine Auswahl an Funktionen aus dem Package angeboten. Sobald die Funktion ausgewählt oder ausgeschrieben ist, zeigt Shift + Tabulator die Signatur der Funktion an. Mit einer Funktionssignatur ist gemeint, wie genau die Funktion definiert ist und welche

Parameter sie unterstützt. Dazu werden direkt die Erläuterungen aus der Dokumentation angezeigt. Bevor Sie den nächsten Absatz lesen, versuchen Sie herauszufinden, wozu der Parameter `sep` verwendet wird.

Da die Werte in der CSV-Datei mit einem Semikolon `;` getrennt sind, muss dieses als sogenannter Separator über `sep=';'` angegeben werden. Läge die Datei im Excel-Format vor, könnte sie über den folgenden Befehl in Python geladen werden:

```
df = pd.read_excel(dateiname)
```

Hierbei müsste das Objekt `dateiname` vorher natürlich definiert oder durch eine Zeichenkette ersetzt werden. Die Pfadangaben beziehen sich immer auf das aktuelle Arbeitsverzeichnis, das ist dasjenige Verzeichnis, in dem das Python-Skript ausgeführt wird.[36]

Um das Objekt `df` anzeigen und damit die eingelesenen Daten betrachten zu können, nutzen Sie den Befehl:

```
display(df)
```

Die resultierenden Dataframes sind Objekte mit eigenen Funktionen, die nach Angabe eines Punktes aufgerufen werden können. Auf diese Weise lassen sich Datensätze auch wieder als CSV- oder Excel-Datei abspeichern:

```
df.to_csv(dateiname, index=False)

df.to_excel(dateiname, index=False)
```

Der Parameter `index=False` sorgt dafür, dass nur die Spalten abgespeichert werden und nicht auch die Namen der Zeilen (siehe oben).

Statt Daten einzulesen lassen sich leere Dataframes erstellen, die dann im Verlauf eines Skripts mit Daten gefüllt werden. Es empfiehlt sich dabei, zunächst ein Dict `{ }` je Datensatz anzulegen, die Dicts in eine Liste `[ ]` zu legen und diese Liste in einen Dataframe umzuwandeln. Der bestehende und der neue Dataframe werden dann zusammengefügt:

---

[36] Das Arbeitsverzeichnis lässt sich nach `import os` mit der Funktion `os.getcwd()` ausgeben und mit `os.chdir()` ändern.

```
data = pd.DataFrame()

data_new = pd.DataFrame([
    {'id':1,'name':'rey'},
    {'id':2,'name':'han'}
])

data = pd.concat(
    [data, data_new],
    sort=False,
    ignore_index=True
)
```

Wird das neu erstellte Objekt nicht mehr benötigt, wird es über den Befehl `del data_new` wieder entfernt.

Spalten und Zeilen filtern Für die Datenanalyse sind neben dem Einlesen und Speichern drei Verfahren besonders wichtig: die Auswahl von Daten, das Berechnen neuer Daten und das Aggregieren von Daten. Im ersten Schritt werden Daten auf die konkreten Bedürfnisse zugeschnitten. Indem Spalten und Zeilen ausgewählt werden, entstehen Teildatensätze. Im Englischen findet man dafür häufig die Begriffe *subsetting*, *indexing* und *slicing*.

Sowohl Spalten als auch Zeilen können entweder über ihre Position oder über ihre Namen angesprochen werden.[37] Die Funktion `loc[]` dient zum Auswählen über die Namen. Der erste Parameter gibt die Zeilen an, der auf das Komma folgende Parameter bestimmt die Spalten. Beachten Sie, dass hier ausnahmsweise eckige Klammern und keine runden Klammern verwendet werden:[38]

```
df.loc[zeilen, spalten]
```

[37] Eine umfassendere Einführung finden Sie bei Petrou (2017; https://medium.com/dunder-data/selecting-subsets-of-data-in-pandas-6fcd0170be9c).

[38] Ein Dataframe enthält die sogenannten magischen Methoden `__setitem__` und `__getitem__`, auf diese werden die eckigen Klammern umgelenkt. Magische Methoden vereinfachen die Arbeit mit Objekten.

An die Stelle der Zeilen- und Spaltenparameter können verschiedene Varianten treten. Wird einer der Parameter nicht verwendet, setzt man einen Doppelpunkt ein, um darüber alle Zeilen oder Spalten auszuwählen (☛ *Repositorium*):

- Ein **einzelner Name einer Spalte** wird in einfache Anführungszeichen gesetzt:

```
df.loc[:, 'replies']
```

- **Mehrere Namen** werden zur Auswahl in einer Liste angegeben:

```
df.loc[:, ['replies', 'favorites']]
```

- Ein **Bereich** von Spalten wird angegeben, indem der Startpunkt vor und der Endpunkt nach einem Doppelpunkt angegeben werden:

```
df.loc[:, 'favorites':'retweets']
```

Startpunkt und Endpunkt vor bzw. nach dem Doppelpunkt können wegfallen, um einen Bereich von Anfang an oder bis an das Ende zu benennen (z. B. `'favorites':` oder `:'retweets'`). Für den Zugriff auf die Spalten, wenn ohnehin alle Zeilen erhalten bleiben sollen, kann zudem die `loc[]`-Funktion ausgelassen werden: `df[spalten]`. Anstelle der Spaltenauswahl können wieder die genannten Optionen treten, das heißt einzelne Spalten, eine Liste mit Spalten oder ein Bereich. Soll nur eine einzelne Spalte ausgewählt werden, kann auf die Dot-Notation zurückgegriffen werden, da Spalten Eigenschaften eines Dataframes sind (z. B. `df.favorites`). Probieren Sie die verschiedenen Varianten einmal aus!

Etwas verwirrend ist zunächst, dass die Auswahl von Spalten über die `loc[]`-Funktion mal einen Dataframe und mal ein Series-Objekt zurückgibt. Wenn die Spalten in eckigen Klammern (also als Liste) angegeben werden, entsteht ein Dataframe, andernfalls eine Series. Sie können mit dem `type()`-Befehl überprüfen, in welcher Form das Ergebnis vorliegt:

```
x = df.loc[:, ['favorites']]
type(x)

x = df.loc[:, 'favorites']
type(x)
```

Ganz allgemein ist die Typüberprüfung hilfreich, wenn etwas nicht wie gewünscht funktioniert. Der Unterschied zwischen Dataframes und Series wird später wichtig, wenn mit den Daten weitergearbeitet wird, denn beide Typen erlauben unterschiedliche Operationen. Achten Sie bei allen besprochenen Funktionen auf die eckigen Klammern – mit runden Klammern gibt es eine Fehlermeldung. Und überlegen Sie stets, welchen Typ das Ergebnis haben müsste – Series oder Dataframe.

Damit nicht nur die Spalten, sondern auch Zeilen mit der `loc[]`-Function über Namen ausgewählt werden können, müssen die Zeilen zunächst Bezeichnungen erhalten. Dazu wird der Zeilenindex angepasst. Es können bestehende Spalten als Index und damit als Quelle der Zeilenbezeichnungen genutzt werden, zum Beispiel die Spalte `from`:

```
df = df.set_index(['from'])
```

Nach der Indexierung der Spalte `from`, können Inhalte aus dieser Spalte in Anführungszeichen gesetzt und als erster Parameter in die `loc[]`-Funktion gegeben werden. Zurückgegeben wird im Beispiel ein neuer Dataframe, der alle Zeilen mit dem Wert „theeduni“ in der zuvor indexierten Spalte `from` enthält:

```
df.loc['theeduni']
```

Eine Angabe von Spalten als zweitem Parameter ist nur erforderlich, wenn nicht alle Spalten zurückgegeben werden sollen. In folgendem Beispiel werden die Zeilen mit den indizierten Namen „theeduni“ und „unialdera“ ausgewählt, gleichzeitig wird auf die Spalte `favorites` eingegrenzt:

```
df.loc[['theeduni', 'unialdera'], ['favorites']]
```

Der Index kann jederzeit zurückgesetzt werden, wobei bestehende Daten in Spalten überführt und alle Zeilen neu nummeriert werden:

```
df = df.reset_index()
```

Will man ohne die Bezeichnungen der Spalten und Zeilen arbeiten, kann mit der `iloc[]`-Funktion (i = integer) die Auswahl über die Position von Spalten und

Tab. 5.7 Bedingungen zum Filtern von Datensätzen

Bedingung	Erläuterung
`df.favorites > 10`	Wert größer als 10
`df.favorites == 10`	Wert gleich 10
`df.favorites != 10`	Wert ungleich 10
`df.favorites <= 10`	Wert kleinergleich 10
`df.favorites.isin([5,10])`	Wert 5 oder 10
`~df.favorites.isin([5,10])`	Wert nicht 5 oder 10
`df.favorites.notnull()`	Wert nicht leer (kein `NaN`)
`df.hashtags.str.contains("tierwelt", na=False)`	Wert enthält „tierwelt", für leere Werte wird `False` ausgegeben (`na`-Parameter)

Quelle: Eigene Darstellung

Zeilen erreicht werden. Das folgende Beispiel wählt die ersten fünf Zeilen und alle Spalten nach der zweiten Spalte aus:[39]

```
df.iloc[:5, 2:]
```

Zeilen mit Bedingungen filtern Häufig müssen Daten nicht nur aufgrund der Spalten- und Zeilennamen ausgewählt, sondern auf Grundlage der Werte gefiltert werden. Dazu formuliert man eine Bedingung und setzt diese in eckige Klammern (siehe Tab. 5.7). So könnte es interessant sein, nur Fälle mit mehr als zehn Favorites auszuwählen, das heißt Fälle mit einem Wert größer als 10 in der entsprechenden Spalte:

```
df[df.favorites > 10]
```

Mehrere Bedingungen können durch `&` (und) sowie `|` (oder) kombiniert werden:

```
df[(df.favorites > 10) & (df.retweets > 5)]
```

In allen Fällen gilt, dass der bestehende Dataframe nicht verändert wird, sondern ein neues Objekt erzeugt wird. Will man damit weiterarbeiten, muss das Ergebnis der Auswahlfunktionen im bestehenden oder einem neuen Objekt abgespei-

[39] Die Zählung von Zeilen und Spalten beginnt bei 0. Da die zweite Angabe im Bereich `:5` nicht inklusiv ist, werden dennoch die ersten fünf und nicht sechs Zeilen ausgewählt.

chert werden. Das folgende Beispiel überschreibt das Objekt `df`, sodass nur noch die ausgewählten Zeilen übrigbleiben:

```
df = df[df.favorites > 10]
```

Der Punkt wird in Python immer verwendet, um auf Eigenschaften und Methoden von Objekten zuzugreifen. Die Spalten verfügen über eine Auswahl an datentypspezifischen Zugriffsmöglichkeiten (siehe unten, dtype specific accessors). Das vereinfacht zum Beispiel die Arbeit mit Textwerten. Textspalten stellen über die Eigenschaft `str` unter anderem Funktionen zum Suchen und Ersetzen oder zur Umformung zwischen Groß- und Kleinschreibung bereit.[40]

Die Funktion `str.contains()` kann zum Beispiel dabei helfen, reguläre Ausdrücke (siehe Abschn. 4.1.1) als Filterbedingung einzusetzen. Die Hilfe zur Funktion können Sie direkt in JupyterLab mit einem Fragezeichen am Ende des Befehls aufschlagen:

```
df.hashtags.str.contains?
```

Die Funktion gibt eine Series mit `True` bzw. `False` für jede passende Zeile zurück:

```
df.hashtags.str.contains(
    "tierwelt|sumpfschnecke|reptilien",
    case=False, regex=True, na=False
)
```

Der reguläre Ausdruck `"tierwelt|sumpfschnecke|reptilien"` bedeutet, dass eines der drei angegebenen Wörter enthalten sein muss. Der Parameter `case=False` sorgt dafür, dass Groß- und Kleinschreibung keine Rolle spielen. Durch die zusätzliche Angabe von `regex=True`, wird das Muster als regulärer Ausdruck behandelt. Mit `na=False` wird festgelegt, dass fehlende Werte nicht als Treffer zählen sollen, also `False` zurückgegeben werden soll, wenn in der Spalte `hashtags` kein Wert steht.

[40] Eine Übersicht über String-Funktionen finden Sie in der *pandas*-Dokumentation, siehe zum Beispiel ganz unten unter Pandas development team (2022c; https://pandas.pydata.org/pandas-docs/stable/user_guide/text.html).

Solche Series mit Wahrheitswerten können genauso wie die oben besprochenen Möglichkeiten als Filterbedingung zur Auswahl von Zeilen eingesetzt werden. Innerhalb der über `str.contains()` erzeugten Series ist für jede Zeile mit `True` oder `False` angegeben, ob sie ausgewählt werden soll (sogenanntes boolean indexing):

```
df[df.hashtags.str.contains( \
   "tierwelt|sumpfschnecke|reptilien", \
   case=False, regex=True, na=False)]
```

Komplexere Filtermöglichkeiten ergeben sich daraus, dass *pandas*-Funktionen die Objekte nicht direkt verändern, sondern immer neue Objekte zurückgeben. Gibt eine Funktion einen Dataframe zurück, verfügt dieser weiterhin über Funktionen zum Auswählen von Spalten oder Zeilen. Dadurch können mehrere Funktionen mit einem Punkt direkt aneinandergekettet werden. Zurückgegeben wird dabei nur das Endergebnis:

```
df_subset = df[df.favorites > 10] \
            .loc[:, ['replies', 'favorites']]
```

Hierbei werden nur Fälle mit mehr als 10 Favorites ausgewählt und durch die angehängte `loc[]`-Funktion bleiben lediglich die Spalten `replies` und `favorites` erhalten.

Neue Spalten erstellen und Werte berechnen Mitunter müssen neue Spalten erstellt oder bestehende Spalten verändert werden, um die Daten aufzubereiten. Stellen Sie sich etwa vor, es soll eine neue Spalte gebildet werden, in der Retweets und Replies verrechnet sind. Spalten mit Zahlen können direkt mit mathematischen Operationen behandelt werden, etwa um sie zu addieren:

```
df['reactions'] = df['retweets'] + df['replies']
```

Dabei muss man beachten, dass die Addition fehlender Werte wiederum fehlende Werte erzeugt. Mit der Funktion `fillna()` lassen sich fehlende Werte vor der Berechnung ersetzen:

```
df['retweets'] = df['retweets'].fillna(0)
```

Weitere mathematische Funktionen wie den Logarithmus oder das Wurzelziehen finden Sie in der Bibliothek *numpy*.

Das Verändern der Spalten funktioniert in den Beispielen so, dass zunächst mit den Funktionen (Addition, fehlende Werte ersetzen) eine neue Liste bzw. ein neues Series-Objekt erzeugt wird. Diese Liste wird dann mit dem Zuweisungsoperator einer Spalte mit dem gewünschten Namen zugeordnet. Gibt es die Spalte noch nicht, wird sie automatisch angelegt.[41] Die im vorangegangenen Abschnitt besprochenen Bedingungen erzeugen solche Listen. Zum Beispiel kann in einer eigenen Spalte namens `tierwelt` festgehalten werden, auf welche Datensätze das Suchmuster zutrifft:

```
df['tierwelt'] = df['hashtags'].str. \
      contains("tierwelt|sumpfschnecke|reptilien", \
      case=False, regex=True)
```

Eine solche neue Spalte kann dann, wie im folgenden Abschnitt beschrieben, weiterverarbeitet werden. So ließe sich etwa auszählen, wie viele Tweets ein tierbezogenes Hashtag enthalten – eine erste, einfache Form automatisierter Textanalyse (siehe Kap. 9).

Aggregieren und auszählen Datenanalyse basiert im Wesentlichen darauf, umfangreiche Daten sinnvoll zusammenzufassen.[42] Je nach Art der Daten sind unterschiedliche Zusammenfassungen sinnvoll. Besonders für textbasierte Kategorien, im folgenden Beispiel die Namen der Tweet-Autor:innen, ist das Auszählen von Werten hilfreich:

```
df['from'].value_counts()
```

Kreuztabellen helfen dabei, die Kombination von Werten zu erfassen, zum Beispiel wer wie viele Tweets mit tierbezogenen Hashtags verwendet. Die Funktion `crosstab()` nimmt dafür als Parameter mehrere zu kombinierende Spalten entgegen:

```
pd.crosstab(df['from'], df['tierwelt'])
```

[41] Weitere Möglichkeiten sind die Funktionen `pd.assign()` und `pd.insert()`, siehe die Dokumentation (Pandas development team 2022b; https://pandas.pydata.org/pandas-docs/stable/reference/frame.html).

[42] Zu unterscheiden sind hierbei deskriptive Zusammenfassungen von modellbasierten Analysen. Erstere beschreiben die Daten (z. B. Mittelwerte), letztere beschreiben den datengenerierenden Prozess (z. B. Regression).

Zahlenbasierte, metrische Werte lassen sich dagegen gut über den Mittelwert anhand der Funktion `mean()` zusammenfassen:

```
df['favorites'].mean()
```

Diese Berechnung lässt sich für mehrere oder alle Spalten eines Dataframes gleichzeitig durchführen:

```
df[['retweets', 'replies']].mean()
```

Grundsätzlich gibt es hier zwei Richtungen, die verrechnet werden können, die Spalten oder die Zeilen. Diese beiden Richtungen werden als Achsen (engl. *axis*) bezeichnet. Normalerweise wird der Mittelwert über alle Zeilen berechnet, diese Achse hat die Nummer 0 und muss nicht extra angegeben werden. Soll eine spaltenübergreifende Berechnung durchgeführt werden, muss als Achsennummer die 1 angegeben werden. Das folgende Beispiel berechnet den Mittelwert je Zeile:

```
df[['retweets', 'replies']].mean(axis=1)
```

Eine schnelle Übersicht über verschiedene Kennwerte ergibt die `describe()`, neben dem Mittelwert werden unter anderem Standardabweichung, Minimum und Maximum ausgegeben. Hier können wie im vorangegangenen Beispiel vorher relevante Spalten ausgewählt werden, oder es werden einfach alle zahlenbasierten Spalten zusammengefasst:

```
df.describe()
```

Um verschiedene Gruppen vergleichen zu können, wird die Funktion `groupby()` eingesetzt. Die Anzahl der Fälle je Gruppe kann zunächst in Kombination mit der `size()`-Funktion ermittelt werden:

```
df.groupby('from').size()
```

Das leistet das Gleiche wie die oben besprochene `value_counts()`-Funktion. Zwischen den Gruppen können aber bei numerischen Werten auch Mittelwerte verglichen werden, etwa in Bezug auf die Retweets und Replies getrennt nach Autor:innen, dazu werden nach der Gruppierung die gewünschten Spalten ausgewählt:

```
df.groupby('from')[['retweets', 'replies']].mean()
```

Auch die `describe()`-Funktion lässt sich auf gruppierte Daten anwenden. Damit es nicht zu unübersichtlich wird, bietet es sich an, nur einzelne Spalten zu fokussieren:

```
df.groupby('from')['favorites'].describe()
```

Das Ergebnis dieser Auswertungen sind selbst wiederum Dataframes, die sich in eigenen Objekten ablegen und als CSV-Datei speichern lassen. Da die Gruppierungsvariable als Index verwendet wird, sollte der Index vorher mit `reset_index()` wieder in eine Spalte überführt werden:

```
df_fav = df.groupby('from')['favorites'].mean()
df_fav = df_fav.reset_index()
df_fav.to_csv("favorites_mean.csv", index=False)
```

Die bislang besprochenen Funktionen `mean()` und `describe()` für metrische sowie `size()` und `value_counts()` für kategoriale Daten sind in die *pandas*-Objekte eingebaut und stehen über die Dot-Notation als Methoden der Series oder Dataframes zur Verfügung. Damit lassen sich bereits erste Analysen durchführen. Für komplexere Aggregationen können die Zeilen und Spalten mit eigenen Funktionen bearbeitet werden. Mit der `apply()`-Funktion lässt sich eine beliebige, auch eine selbst geschriebene, Funktion auf eine der Achsen eines Dataframes anwenden. Ähnlich arbeitet `transform()`, aber mit genaueren Anforderungen an die Parameter und Rückgabewerte. Durch die `agg()`-Funktion lassen sich gleich mehrere Funktionen achsenweise ausführen. Beispiele und Erläuterungen finden Sie in der Dokumentation von *pandas*. Dort werden Sie auch auf eine besondere Art stoßen, benutzerdefinierte Funktionen zu definieren: Der `lambda`-Operator erlaubt es, sehr kompakt sogenannte anonyme Funktionen zu definieren, um sie in Kombination mit den Aggregationsfunktionen einzusetzen.

5.2.4 Wie geht es weiter?

Der hier beschriebene Einstieg über Jupyter Notebooks ist ein typischer Weg, um mit Python loszuentwickeln. Werden die Projekte nach und nach größer, die Dateien wachsen an und der Code wird komplexer, dann lohnt sich die Beschäftigung mit weiteren Entwicklungsumgebungen. In wissenschaftlichen Projekten wird zum Beispiel die kostenfreie Entwicklungsumgebung Spyder genutzt. Eine kommerzielle Entwicklungsumgebung mit umfangreichen Funktionen ist PyCharm.

Für Open-Source-Projekte vergibt der Anbieter auch kostenlose Lizenzen. PyCharm kann sowohl für die wissenschaftliche Datenanalyse als auch für klassische Anwendungsentwicklung eingesetzt werden. Beide Umgebungen sind mächtiger als JupyterLab, erfordern aber auch eine etwas längere Einarbeitung. Dafür enthalten sie viele nützliche Funktionen – wer mit R und RStudio gearbeitet hat, vermisst in Notebooks möglicherweise die Option, Datenobjekte auch ohne `print()` -Befehle anzusehen. Geboten werden zudem Funktionen zum Debugging – das heißt zur Fehlersuche und -behebung (siehe Abschn. 6.2).

Übungsfragen

1. Was ist JupyterLab und was sind Jupyter Notebooks?
2. Wozu dient pip?
3. Mit welchem Python-Befehl können Ergebnisse angezeigt werden?
4. Wie lauten in Python die Muster für folgende Sprachelemente?
   a) Schleifen
   b) Wenn-Dann-Verzweigungen
   c) Funktionsdefinitionen
5. Wozu wird das Package *pandas* eingesetzt?
6. Welche Möglichkeiten fallen Ihnen ein, um mit Python Spalten und Zeilen eines Datensatzes auszuwählen?
7. Wie kann mit Python die Anzahl der Fälle in einem Datensatz festgestellt werden?
8. Wie kann mit Python für eine Spalte der Mittelwert berechnet werden?

Weiterführende Literatur

Es gibt unzählige Einführungen in Python. Einige sind eher allgemein ausgelegt, andere spezifisch für sozial- und geisteswissenschaftliche Analysen. Einige sind kostenlos online nutzbar, andere ausschließlich in Papierform zu erwerben. Einige richten sich an Einsteiger:innen, andere an fortgeschrittene, erfahrene Programmierer:innen. In der folgenden Liste finden Sie für jede dieser Kategorien mindestens einen Titel.

Brooker, P. D. (2019). *Programming with Python for Social Scientists*. Los Angeles: Sage.

Downey, A. B., Fröhlich, S. & Gherman, D. C. (2014). *Programmieren lernen mit Python: Einstieg in die Programmierung* (2. Aufl.). Beijing: O'Reilly.

Ernesti, J. & Kaiser, P. (2018). *Python 3: Das umfassende Handbuch* (5., aktualisierte Aufl.). Bonn: Rheinwerk Verlag.

Lutz, M. (2007). *Einführung in Python* (2. Aufl.). Köln: O'Reilly Verlag.

Shaw, Z. (2014). *Learn Python the hard way: A very simple introduction to the terrifyingly beautiful world of computers and code* (3. Aufl.). Upper Saddle River: Addison-Wesley.

Steyer, R. (2018). *Programmierung in Python: Ein kompakter Einstieg für die Praxis*. Wiesbaden: Springer Vieweg.

Vanderplas, J. T. (2016). *Python data science handbook: Essential tools for working with data*. Sebastopol: O'Reilly. Online unter: https://tanthiamhuat.files.wordpress.com/2018/04/pythondatasciencehandbook.pdf

Literatur

Anaconda. (2020). Anaconda Software Distribution [Computer software]. https://docs.anaconda.com/

Chambers, J. M. (2014). Object-Oriented Programming, Functional Programming and R. *Statistical Science*, *29*(2), 167–180. https://doi.org/10.1214/13-STS452

CRAN. (2022). *The Comprehensive R Archive Network*. Zugriff am 24.04.2022. https://cran.r-project.org/

Dahley, D. (2017). *If statistics programs/languages were cars…* Zugriff am 11.07.2022. https://twitter.com/statsepi/status/872343239931682816

Field, A., Miles, J. & Field, Z. (2012). *Discovering statistics using R*. Los Angeles: Sage.

GitHub. (2022a). Atom (Version 1.60.0) [Computer software]. https://atom.io/

Guy_Jantic. (2019). *If statistics programs/languages were cars…* Zugriff am 24.04.2022. https://www.reddit.com/r/rstats/comments/dn7fa7/i_thought_the_if_stats_programs_were_cars_meme/

Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D. [David] et al. (2020). Array programming with NumPy. *Nature*, *585*(7825), 357–362. https://doi.org/10.1038/s41586-020-2649-2.

Ho, D. (2022). Notepad++ (Version 8.4.1) [Computer software]. https://notepad-plus-plus.org/

Holtz, Y. (2018). *The R Graph Gallery*. Zugriff am 24.04.2022. https://r-graph-gallery.com/

Hunter, J., Dale, D., Firing, E., Droettboom, M. & Matplotlib development team. (2022). Matplotlib: Visualization with Python (Version 3.5.2) [Computer software]. https://matplotlib.org/

Inkscape. (2022). Open Source Scalable Vector Graphics Editor (Version 1.1.2) [Computer software]. https://inkscape.org/

Jeppson, H., Hofmann, H., Di Cook & Wickham, H. (2021). ggmosaic: Mosaic Plots in the 'ggplot2' Framework (Version 0.3.3) [Computer software]. https://cran.r-project.org/package=ggmosaic

JetBrains. (2022a). PyCharm. The Python IDE for Professional Developers (Version 2022.1.3) [Computer software]. www.jetbrains.com/pycharm

Jupyter. (2022). JupyterLab. A Next-Generation Notebook Interface [Computer software]. https://jupyter.org/

Kabacoff, R. I. (2017). *Quick-R*. Zugriff am 24.04.2022. https://www.statmethods.net/

McPherson, J. (2021). *Debugging with the RStudio IDE*. Zugriff am 24.04.2022. https://support.rstudio.com/hc/en-us/articles/205612627-Debugging-with-RStudio

Müller, K., Wickham, H. & R Special Interest Group on Databases. (2022a). DBI: R Database Interface (Version 1.1.3) [Computer software]. https://cran.r-project.org/package=DBI

Müller, K., Wickham, H., Francois, R. & Bryan, J. (2022b). tibble: Simple Data Frames (Version 3.1.7) [Computer software]. https://cran.r-project.org/package=tibble

Newman, M. E. (2005). Power laws, Pareto distributions and Zipf's law. *Contemporary Physics*, *46*(5), 323–351. https://doi.org/10.1080/00107510500052444

Ooms, J. & McNamara, J. (2021). writexl. Export Data Frames to Excel 'xlsx' Format (Version 1.4.0) [Computer software]. https://cran.r-project.org/package=writexl

Pandas development team. (2022a). Pandas (Version 1.4.3) [Computer software]. *Zenodo*. https://doi.org/10.5281/zenodo.3509134

Pandas development team. (2022b). *DataFrame*. Zugriff am 25.04.2022. https://pandas.pydata.org/pandas-docs/stable/reference/frame.html

Pandas development team. (2022c). *Working with text data*. Zugriff am 25.04.2022. https://pandas.pydata.org/pandas-docs/stable/user_guide/text.html

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O. et al. (2011). Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, *12*, 2825–2830.

Petrou, T. (2017). Selecting Subsets of Data in Pandas. *Medium*. Zugriff am 25.04.2022. https://medium.com/dunder-data/selecting-subsets-of-data-in-pandas-6fcd0170be9c

Python Software Foundation. (2019a). *Python Success Stories*. Zugriff am 24.12.2019. https://www.python.org/about/success/ilm/

Python Software Foundation. (2019b). *General Python FAQ*. Zugriff am 24.12.2019. https://docs.python.org/3/faq/general.html#why-is-it-called-python

Python Software Foundation. (2022b). *Built-in Types. Text Sequence Type*. Zugriff am 25.04.2022. https://docs.python.org/3/library/stdtypes.html#text-sequence-type-str

Python Software Foundation. (2022c). Python (Version 3.10.5) [Computer software]. https://docs.python.org/3/index.html

Python Software Foundation. (2022d). *Virtual Environments and Packages*. Zugriff am 25.04.2022. https://docs.python.org/3/tutorial/venv.html

R Core Team. (2021). *R Language Definition. Version 4.1.2*. Zugriff am 25.04.2022. https://cran.r-project.org/doc/manuals/r-release/R-lang.pdf

R Core Team. (2022). *The R Project for Statistical Computing. R version 4.2.0*. Zugriff am 24.04.2022. https://www.r-project.org/

Reitz, K. (2021). *The Hitchhiker's Guide to Python*. Zugriff am 25.04.2022. https://docs.python-guide.org/

RStudio. (2022a). *Download the RStudio IDE*. Zugriff am 25.04.2022. https://www.rstudio.com/products/rstudio/download/

RStudio (2022b). *RStudio Cheatsheets*. Zugriff am 24.04.2022. https://www.rstudio.com/resources/cheatsheets/

Spinu, V., Grolemund, G., Wickham, H., Vaughan, D., Lyttle, I., Costigan, I. et al. (2021). lubridate: Make Dealing with Dates a Little Easier (Version 1.8.0) [Computer software]. https://cran.r-project.org/package=lubridate

Spyder project contributors. (2022). Spyder. *The Scientific Python Development Environment* (Version 531) [Computer software]. https://www.spyder-ide.org/

Stack Overflow. (2022). *A public platform building the definitive collection of coding questions & answers*. Zugriff am 24.04.2022. https://stackoverflow.com

The pip developers. (2022). pip. The package installer for Python [Computer software]. https://pip.pypa.io/

W3Schools. (2022b). *Python Classes and Objects*. Zugriff am 25.04.2022. https://www.w3schools.com/python/python_classes.asp

W3Schools. (2022c). *Python Try Except*. Zugriff am 25.04.2022. https://www.w3schools.com/python/python_try_except.asp

Waring, E., Quinn, M., McNamara, A., Arino de la Rubia, E, Zhu, H. & Ellis, S. (2022). skimr. Compact and Flexible Summaries of Data [Computer software]. https://docs.ropensci.org/skimr/

Wickham, H. (2019a). stringr. Simple, Consistent Wrappers for Common String Operations (Version 1.4.0) [Computer software]. https://cran.r-project.org/package=stringr

Wickham, H. (2019b). *Tidyverse. R packages for data science*. Zugriff am 24.04.2022. https://www.tidyverse.org/

Wickham, H. (2021). *The tidyverse style guide*. Zugriff am 24.04.2022. https://style.tidyverse.org/

Wickham, H. & Girlich, M. (2022). tidyr: Tidy Messy Data (Version 1.2.0) [Computer software]. https://cran.r-project.org/package=tidyr

Wickham, H. & Grolemund, G. (2016). *Data visualisation*. Zugriff am 24.04.2022. https://r4ds.had.co.nz/data-visualisation.html

Wickham, H., Girlich, M. & Ruiz, E. (2022). dbplyr. A 'dplyr' back end for databases [Computer software]. https://dbplyr.tidyverse.org/

Wickham, H., Miller, E. & Smith, D. (2022). haven: Import and Export 'SPSS', 'Stata' and 'SAS' Files (Version 2.5.0) [Computer software]. https://cran.r-project.org/package=haven

Wickham, H., François, R., Henry, L. & Müller, K. (2022a). dplyr: A Grammar of Data Manipulation (Version 1.0.9) [Computer software]. https://cran.r-project.org/package=dplyr

Wickham, H., Bryan, J., Kalicinski, M., Valery, K., Leitienne, C., Colbert, B. et al. (2022b). readxl: Read Excel Files (Version 1.4.0) [Computer software]. https://cran.r-project.org/package=readxl

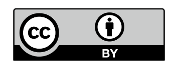
Wickham, H., Chang, W., Henry, L., Lin Pederson, T., Takahashi, K., Wilke, C. et al. (2022c). *Function reference ggplot2*. Zugriff am 24.04.2022. https://ggplot2.tidyverse.org/reference/

Wickham, H., Chang, W., Henry, L., Lin Pederson, T., Takahashi, K., Wilke, C. et al. (2022d). *Learning ggplot2*. Zugriff am 24.04.2022. https://ggplot2.tidyverse.org/#learning-ggplot2

Wickham, H., Hester, J., Francois, R., Bryan, J., Bearrows, S., Jylänki, J. et al. (2022e). readr: Read Rectangular Text Data (Version 2.1.2) [Computer software]. https://cran.r-project.org/package=readr

Wilkinson, L. (1999). *The grammar of graphics*. New York: Springer.

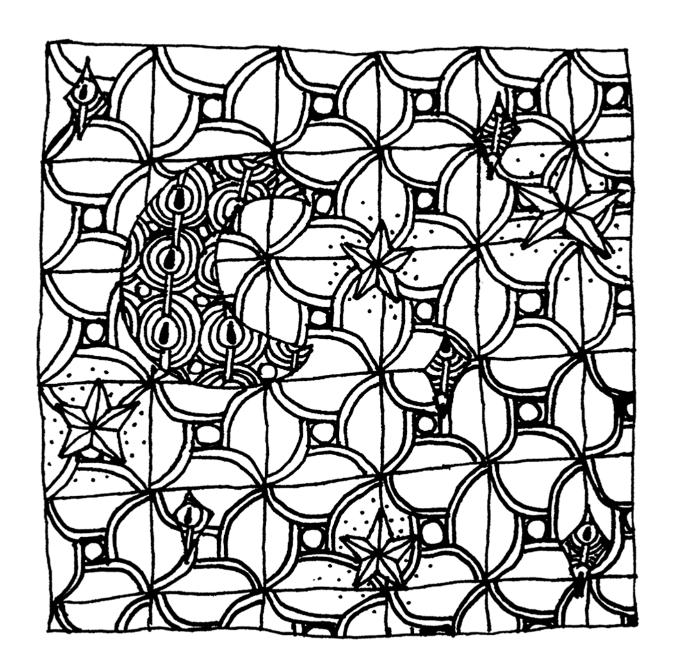
WinPython development team. (2022). WinPython [Computer software]. https://winpython.github.io/


BY



Scaling up. Daten und Skripte organisieren





J. Jünger, C. Gärtner, *Computational Methods für die Sozial- und Geisteswissenschaften*, https://doi.org/10.1007/978-3-658-37747-2_6

Zusammenfassung

Dieses Kapitel führt in Konzepte ein, die bei der Koordination von größeren Projekten und der Zusammenarbeit mit anderen helfen. Sie lernen, was eine Versionsverwaltung ist und worauf Sie bei der Wahl von Entwicklungsumgebungen achten können. In Bezug auf rechenintensivere Projekte finden Sie eine Einführung in die Virtualisierung und erhalten einen Einblick in High-Performance-Computing.

Im Online-Repositorium unter https://github.com/strohne/cm finden Sie begleitend zum Kapitel weitere Materialien, auf die wir im Text mit ☛ verweisen.

Schlüsselwörter

Versionsverwaltung · Git · Entwicklungsumgebung · IDE · Debugger · Coding Style · Webserver · LAMP · Vagrant · Docker · Virtualisierung · Dashboards · Parallelisierung · Cloud Computing · High Performance Computing · Slurm

Die ersten Schritte der automatisierten Datenerhebung und -analyse lassen sich noch gut in einzelnen Skripten auf dem eigenen Computer umsetzen. Wenn Projekte über einen längeren Zeitraum laufen oder zwischen verschiedenen Beteiligten verteilt sind, sammeln sich jedoch schnell umfangreiche Datenbestände und Skripte an. Bei der Koordination solcher Projekte helfen Versionsverwaltungen wie Git, in denen alle Änderungen erfasst und miteinander abgeglichen werden (Abschn. 6.1). Umfangreicher Code wird im Laufe von Projekten häufig zunehmend modularisiert. Das heißt, aus einzelnen Codeblöcken entstehen in sich geschlossene Bausteine, die meist auf mehrere Skripte mit unterschiedlichen Funktionen aufgeteilt werden. Entwicklungsumgebungen wie RStudio (RStudio 2022a) oder PyCharm (JetBrains 2022a) stellen dabei eine Vielzahl an nützlichen Werkzeugen bereit, um die Übersicht zu behalten, den Code zu dokumentieren und auf Fehlersuche zu gehen (Abschn. 6.2). Damit die entwickelten Skripte auch auf anderen Computern oder Servern im Internet laufen, können Laufzeitumgebungen festgelegt und untereinander geteilt werden, sodass zum Beispiel alle Beteiligten über die gleiche virtuelle Maschine verfügen (Abschn. 6.3). Spätestens wenn Datensätze zu groß oder Datenanalysen zu rechenintensiv für den eigenen Computer werden, führt der Weg schließlich in die Cloud, das heißt ein Projekt wird auf einer verteilten Serverinfrastruktur ausgeführt (Abschn. 6.4).

Die folgenden Kapitel führen kurz in die jeweiligen Techniken zur Organisation von Daten, Skripten und Ressourcen ein. Wenn man sich das erste Mal damit beschäftigt, ist diese Welt ziemlich abstrakt. Beispielsweise werden Dateien, um sie an andere Wissenschaftler:innen weiterzugeben, nicht mehr kopiert, sondern geklont – und sind somit an verschiedenen Orten gleichzeitig aktuell. R- oder Python-Skripte werden nicht mehr auf der Hardware des eigenen Computers ausgeführt, sondern in einer virtuellen Maschine mit einem eigenen Betriebssystem. Das nachfolgende Kapitel bietet keine vollumfängliche Einführung in die jeweiligen Techniken, vielmehr werden diese konzeptionell besprochen. Versuchen Sie sich beim Lesen des Textes bzw. beim Ausführen der Skripte eine kognitive Landkarte aufzubauen und halten Sie sich vor Augen, wie die unterschiedlichen Konzepte zusammenspielen. Am Ende des Kapitels sollten Sie eine Vorstellung davon haben, wie Computational Methods organisiert werden, sobald Projekte an Größe gewinnen.

6.1 Versionsverwaltung

Systeme zur Versionsverwaltung sind hilfreich, um einzelne Arbeitsschritte zu dokumentieren und Zwischenstände zu speichern. Dadurch lassen sich Entwicklungsschritte im Nachhinein leichter nachvollziehen und verschiedene Versionen des Arbeitsstandes können jederzeit wiederhergestellt werden. Auch wenn man gemeinsam an Projekten arbeitet, können über mehrere Computer hinweg Arbeitsstände über die Versionsverwaltung ausgetauscht werden. Wenn mehrere Personen parallel an den gleichen Skripten arbeiten, übernimmt die Versionsverwaltung anschließend das Zusammenführen der unterschiedlichen Versionen.

Verbreitet ist vor allem die Versionsverwaltung Git. Git kann vollständig über die Kommandozeile bedient werden (siehe Abschn. 1.2.2 zur Arbeit mit der Kommandozeile). Dazu installieren Sie zunächst die auf der Download-Seite für Ihr Betriebssystem passende Version.[1] Die Vielfalt der Kommandos kann am Anfang verwirrend sein.[2] Sie müssen die folgenden Konzepte nicht sofort im Detail verstehen, viele dieser Vorgänge erschließen sich nach und nach:

---

[1] Siehe Git (2022a; https://git-scm.com/downloads).

[2] Für die vollständige Referenz siehe Git (2022b; https://git-scm.com/docs).

- **Init:** Zunächst wird ein lokales Repositorium eingerichtet. Das bedeutet, ein Verzeichnis auf dem eigenen Computer wird unter Versionsverwaltung gestellt. Dafür navigieren Sie mit der Kommandozeile zu dem entsprechenden Verzeichnis und führen den Befehl `git init` aus. Dadurch wird ein verstecktes Verzeichnis mit dem Namen *.git* angelegt, in dem die Versionsverwaltung Änderungen speichert. Dieses Verzeichnis sollte nicht angetastet werden, außer in einem Fall: Wenn Sie es löschen, ist die Versionsverwaltung wieder aufgehoben.
- **Commit:** Nachdem Dateien im Arbeitsverzeichnis erstellt und bearbeitet wurden, fügt der Befehl `git add <dateiname>` die anstelle des Platzhalters `<dateiname>` angegebene Datei zur Versionsverwaltung hinzu. Mit `git add .` können auch einfach alle geänderten Dateien aufgenommen werden, beachten Sie den Punkt am Ende des Befehls. Anschließend führt man einen Commit durch. Dieser fixiert den Zwischenstand, wobei nur die geänderten Zeilen in den Dateien gespeichert werden. Jeder Commit bekommt automatisch eine eindeutige Nummer, sodass man später darauf zurückgreifen kann. Außerdem formuliert man eine Commit Message, in der die letzten Änderungen kurz erläutert werden. Auf der Kommandozeile wird dies über `git commit -m <message>` umgesetzt, wobei `<message>` in Anführungszeichen gesetzt und durch eine kurze Erläuterung ersetzt wird.
- **Branch:** Man kann mehrere Versionen parallel bearbeiten, indem man einen Branch erzeugt. Zu Beginn steht lediglich ein main-Branch[3] zur Verfügung. Vor allem vor umfangreichen Änderungen, zum Beispiel bei der Entwicklung einer neuen Funktion, sollte ein neuer Branch angelegt werden: `git branch <name>`. Der Platzhalter `<name>` wird durch einen eigenen Namen ersetzt. Zu diesem neuen Branch kann mit `git checkout <name>` gewechselt werden. Nachdem Änderungen an den Dateien committed sind, kann man mit `git checkout main` wieder zurückwechseln.[4]

[3] Lange Zeit wurde der Hauptzweig „master“ benannt. Um sprachliche Diskriminierung zu verringern, findet momentan ein Umdenken bei vielen in der Programmierwelt gängigen Begriffen statt (Menge-Sonnentag 2021).

[4] Um die gemeinsame Entwicklung in einem Team zu koordinieren, kann man sich an etablierten Workflows orientieren. Ein verbreitetes Entwicklungsmodell ist beispielsweise der Feature-Branch-Workflow (Santacroce 2015, S. 105). Dabei wird für jedes Feature, das entwickelt wird, ein eigener Feature-Branch vom main-Branch abgezweigt. Ist das Feature fertig implementiert, werden die Änderungen in den main-Branch gemerged und der Feature-Branch wird anschließend wieder gelöscht. Übersichtliche Tutorials zur Arbeit mit Git finden sich unter anderem bei Atlassian (2022; https://www.atlassian.com/git/tutorials).

- **Merge:** Ist die Arbeit auf anderen Branches abgeschlossen, können die Änderungen in den main-Branch übertragen werden. Vor dem sogenannten Mergen wechselt man auf den Zielbranch, in der Regel main. Mit git merge <name> wird dann der Stand aus dem mit <name> angegebenen Branch übertragen. Normalerweise ist es kein Problem, wenn zwischenzeitig auch im main-Branch Änderungen vorgenommen werden. Denn die Übertragung erfolgt zeilenweise, das heißt, es werden immer veränderte Zeilen gelöscht und neu eingefügt (Abb. 6.1). Eine Alternative zum Mergen ist das Rebasen, bei dem die Versionsgeschichte so umgeschrieben wird, dass alle Commits eines Branches hinter die Commits des anderen Branches verschoben werden. Bei Änderungen der Versionsgeschichte sind allerdings Absprachen im Entwicklungsteam nötig.
- **Konflikte auflösen:** Nur wenn die gleichen Zeilen geändert wurden, kommt es beim Mergen zu einem Konflikt. In diesem Fall wird das Mergen angehalten und in der betroffenen Datei werden beide Versionen eingetragen. Mit einem Texteditor kann die entsprechende Datei geöffnet werden, um nach Markierungen wie HEAD oder MAIN zu suchen, wobei HEAD die Version im aktuellen lokalen Branch kennzeichnet. Hier muss dann manuell entschieden werden, welche Zeilen beibehalten werden. Die nicht mehr benötigten Zeilen und die Markierungen werden gelöscht und die Datei committed, um den Merge abzuschließen.
- **Status:** Mit git status erhalten Sie einen Überblick über den Zustand des Repositoriums und sehen beispielsweise, ob noch Konflikte vorliegen. Hilfreich sind zudem git diff, um Änderungen der Dateien und git log, um die Versionsgeschichte anzuzeigen.

```
2 src/actions.py
@@ -473,7 +473,7 @@ def queryNodes(self, indexes=False, apimodule=False, options=False)
473 473     progress.showInfo('threads',u"{} active thread(s)".format(threadpool.getThreadCount()))
474 474
475 475     #auto cancel after three consecutive errors, ignore on streaming-tab
476       - if (status == 'fetched (200)') or (status == 'stream'):
    476 + if (status == 'fetched (200)') or (status == 'stream') or (status == 'downloaded (200)'):
477 477         errorcount=0
478 478     else:
479 479         errorcount += 1
```

Abb. 6.1 Beispiel für einen Commit, bei dem eine Zeile verändert wurde. Quelle: Jünger und Keyling (2015; https://github.com/strohne/Facepager/commit/897d000747ec0b4b5d393d0bff86a89a59d617f0#diff-40fe3da5e33280189b65195d7cb0220d))

Versionsverwaltungen ermöglichen vor allem die Zusammenarbeit mit mehreren Personen an den gleichen Skripten. Dazu werden die Zwischenstände auf eine gemeinsame Plattform wie GitHub[5] oder GitLab[6] hochgeladen und miteinander abgeglichen. Insbesondere Open-Source-Programme werden über diese Plattformen koordiniert. Dabei sind vor allem drei Konzepte wichtig:

- **Clone:** Mit `git clone <URL>` wird ein komplettes Repositorium von der Plattform in das lokale Arbeitsverzeichnis, in dem man sich auf der Kommandozeile befindet, übertragen. Hinter dem Befehl kann noch der Name eines Unterverzeichnisses angegeben werden, in den das Repositorium heruntergeladen werden soll. Ohne Angabe wird automatisch der Name des Repositoriums zum Erstellen eines Unterverzeichnisses verwendet. Soll kein Unterverzeichnis entstehen, kann ein Punkt `.` angehängt werden (siehe für ein Beispiel Kap. 1).
- **Push:** Über `git push` werden Änderungen zur ursprünglichen Quelle hochgeladen. Diese Quelle ist das Online-Repositorium, aus dem die Datei geklont wurde. Um darauf zuzugreifen, benötigt man ein Konto, zum Beispiel bei GitHub. Der Besitzer oder die Besitzerin des Repositoriums muss außerdem die entsprechenden Rechte freigeben. Bei neuen Projekten, die noch nicht veröffentlicht wurden, legt man auf dem Server zunächst ein eigenes Repositorium an, fügt die URL zum lokalen Repositorium mittels `git remote add origin <URL>` hinzu und kann anschließend die Änderungen auf den Server pushen.
- **Pull:** Wenn man mit anderen zusammenarbeitet oder von verschiedenen Computern aus arbeitet, kann man sich über `git pull` alle aktuellen Änderungen vom Server herunterladen und lokal einspielen. Dieser Schritt muss auch immer geschehen, bevor man eigene Änderungen hochlädt. Achtung: Änderungen, die noch nicht committed sind, werden durch den `pull`-Befehl überschrieben. Verschaffen Sie sich vorher stets mit `git status` einen Überblick über den Zustand des lokalen Repositoriums.
- **Pull Request:** Will man zu einem Projekt beitragen, zu dem man keinen Schreibzugriff hat, kann man das Repositorium auf den eigenen Computer klonen, verändern und dann einen Pull Request stellen. Der Inhaber oder die Inhaberin des originalen Repositoriums erhält eine Nachricht und kann die Änderungen ggf. übernehmen. Wie dabei vorgegangen wird, ist in der Hilfe der jeweiligen Plattform (zum Beispiel GitHub oder GitLab) beschrieben.

[5] Siehe GitHub (2022b; https://github.com/).

[6] Siehe GitLab (2022; https://about.gitlab.com/).

Es müssen nicht immer alle Dateien in einem Verzeichnis unter Versionsverwaltung gestellt werden und vor allem große binäre Dateien, wenn es sich also nicht um Texte, sondern zum Beispiel um Videos oder Bilder handelt, sollten tendenziell nicht in eine Versionsverwaltung aufgenommen werden. Solche Ausnahmen können in der Datei *.gitignore* festgelegt werden. Wenn Sie auf Plattformen wie GitHub oder GitLab ein Repositorium anlegen, können Sie in der Regel aus entsprechenden Vorlagen auswählen.

Für den Einstieg in die Arbeit mit Versionsverwaltungen ist – zusätzlich zur Installation von Git – ein grafischer Client hilfreich, insbesondere um vor dem Commit und Push die Änderungen noch einmal zu überprüfen oder die Historie der letzten Änderungen nachzuvollziehen. Mit einem solchen Client erlernt man zudem nach und nach die Konzepte und Befehle. Bei der Installation von Git wird in der Regel bereits ein passender Client mitgeliefert. Eine einfache Alternative ist GitHub Desktop[7] (Abb. 6.2).

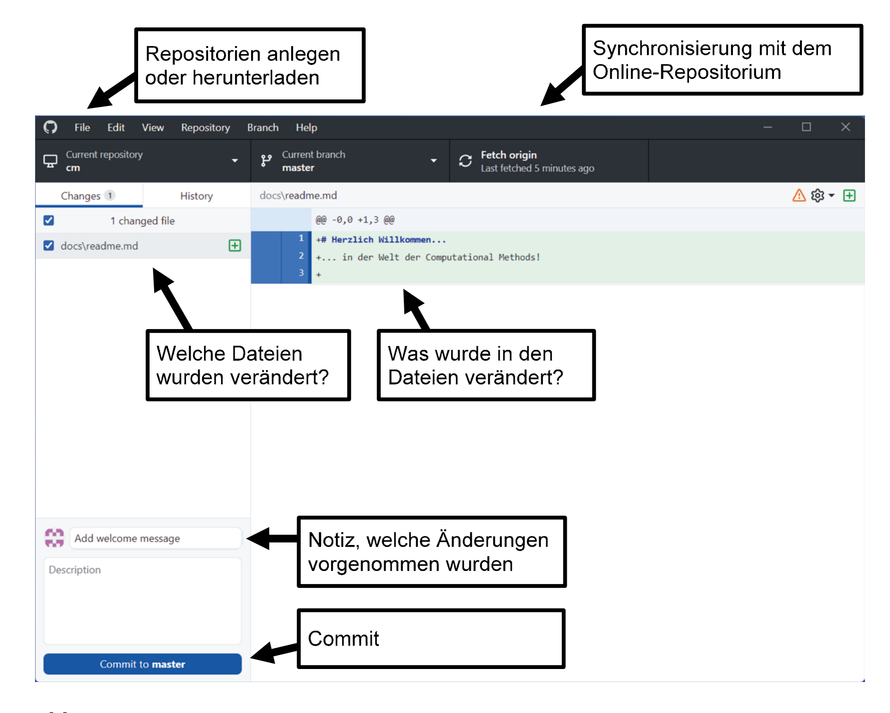


Abb. 6.2 Die wichtigsten Funktionen von GitHub Desktop. (Quelle: eigene Darstellung)

[7] Siehe GitHub (2022c; https://desktop.github.com/).

In GitHub Desktop lässt sich über das FILE-Menü ein neues Repositorium erstellen (NEW REPOSITORY) oder ein vorhandenes Repositorium verwalten (ADD LOCAL REPOSITORY). Außerdem kann ein Repositorium von GitHub oder einer anderen Plattform heruntergeladen werden. Dafür wählt man im FILE-Menü den Punkt CLONE REPOSITORY aus. Dort sind über den Reiter URL anschließend Felder verfügbar, um die Adresse des Repositoriums und das Arbeitsverzeichnis einzutragen. Zuletzt wird das Repositorium über CLONE heruntergeladen und in die lokale Ordnerstruktur eingebunden.

Mit der Schaltfläche FETCH werden das lokale und das Online-Repositorium automatisch durch Pull und Push abgeglichen – die Bezeichnung der Schaltfläche ändert sich entsprechend. Nach dem Bearbeiten von Ordnern und Dateien kann man eigene Änderungen über die Schaltfläche COMMIT eintragen. Dafür beschreibt man zunächst in dem Summary-Feld die Änderungen, bevor diese in den main-Branch committet werden können. Über die Schaltfläche PUSH werden die Änderungen abschließend in das Online-Repositorium übertragen. Falls es dort zwischenzeitig Änderungen gab, muss man vorher die aktuelle Online-Version über PULL herunterladen und mergen, anschließend wird der eigene Stand über erneutes Klicken zur Primärquelle gepusht.

Auch wenn die Vielzahl der Funktionen anfänglich erschlagend sein kann, lohnt sich der Einstieg in Versionsverwaltungen auf lange Sicht. Sie müssen (außer in sehr speziellen Fällen) keine Sorge haben, etwas zu zerstören und können sich nach und nach einarbeiten.

Übungsfragen

1. Was bedeutet Klonen in der Versionsverwaltung Git?
2. Was passiert in einer Versionsverwaltung, wenn zwei Personen gleichzeitig eine Datei ändern?
3. Was ist der Unterschied zwischen den Befehlen `git pull` und `git push`?
4. Mit welchen Befehlen tragen Sie geänderte Dateien in die Versionsverwaltung ein und laden diese in ein Online-Repositorium hoch?

6.2 Entwicklungsumgebungen

Für den Start mit Computational Methods reicht ein einfacher Texteditor aus: In einer Textdatei werden die Anweisungen an den Computer Zeile für Zeile eingegeben, diese Datei wird dann zum Beispiel über die Kommandozeile an R oder

an Python zum Abarbeiten übergeben und das Ergebnis wird am Bildschirm oder in einer neuen Datei ausgegeben. Dieses Input-Throughput-Output-Verfahren setzt voraus, dass man den Programmablauf konzeptionell vordenkt, vollständig verschriftlicht und die Maschine schließlich fehlerfrei arbeitet. Eine solche Perfektion ist allerdings weder realistisch noch erstrebenswert. Denn viele Ideen entwickeln sich erst schrittweise im Verlauf der Analyse, zum Dataminingprozess siehe Kap. 4, und Fehler sind immer einprogrammiert.

Man braucht also eine Möglichkeit, um die Ideen zu organisieren sowie in den Programmablauf und die verarbeiteten Daten hineinzuschauen. Hierbei helfen Integrierte Entwicklungsumgebungen (engl. *Integrated Development Environments*, *IDEs*), die man danach unterscheiden kann, ob sie auf eine bestimmte Programmiersprache beschränkt oder universell einsetzbar sind. Einige Entwicklungsumgebungen sind in den Kapiteln zu R (Abschn. 5.1) und Python (Abschn. 5.2) bereits eingeführt worden. Während RStudio auf R zugeschnitten ist (RStudio 2022a), sind PyCharm (JetBrains 2022a) und Spyder (Spyder 2022) für die Entwicklung mit Python ausgelegt. Alle diese Umgebungen stellen auch Funktionen zum Umgang mit Datenbanken, zur Bedienung von Terminals oder zum Erzeugen von Berichten mit Markdown zur Verfügung. Universeller sind dagegen Umgebungen wie Jupyter Lab (wenn auch meist in Kombination mit Python verwendet; Jupyter 2022), Eclipse (auch wenn es aus der Java-Welt kommt; Eclipse Foundation 2022) oder Visual Studio Code (Microsoft 2022b). Die genannten Entwicklungsumgebungen sind zumindest in den Basisversionen für akademische Zwecke kostenlos erhältlich oder stehen sogar unter Open-Source-Lizenzen.

Diese und andere Entwicklungsumgebungen vereinen eine Vielzahl von Hilfsmitteln unter einem Dach:

- **Projektorganisation:** Damit nicht der gesamte Code in einer einzigen, vermutlich mit der Zeit sehr unübersichtlichen Datei steht, wird er auf mehrere Dateien aufgeteilt. Wenn sich mehrere Dateien aufeinander beziehen, es wird etwa eine Variable aus einer anderen Datei verwendet, dann kann leicht zu den entsprechenden Stellen im Code gesprungen werden. In einer Entwicklungsumgebung werden alle Dateien eines Projekts nicht nur aufgelistet, sondern können auf einmal durchsucht werden, zum Beispiel um dateiübergreifend Variablen umzubenennen. Eine gute IDE unterstützt zudem die Refaktorierung des Codes – darunter sind neben Umbenennungen auch Umstrukturierungen zu verstehen, die zwar keine Änderungen der Funktionalität bewirken, aber den Code eleganter, lesbarer oder besser wartbar machen. Viele IDEs integrieren eine Versionsverwaltung (siehe Abschn. 6.1), mit der die Arbeit im Team erleichtert wird.

- **Dokumentation und Coding Style**: Was einem im Moment der Entwicklung noch selbstverständlich vorkommt, ist einige Wochen später schon wieder kryptisch geworden. Deshalb sind die Dokumentation von Code und ein einheitlicher Coding Style empfehlenswert (Abb. 6.3). Zur Dokumentation von Funktionen in Python haben sich beispielsweise sogenannte Docstrings etabliert, in denen vor jeder Funktion eine Beschreibung, die Input- und die Rückgabewerte festgehalten werden. Ein Coding Style legt zudem fest, welche Konventionen bei der Benennung von Variablen oder der Definition von Funktionen eingehalten werden sollten. IDEs unterstützen dabei, entsprechende Standards einzuhalten, indem sie – wie die automatisierte Rechtschreibprüfung in einer Textverarbeitungssoftware – auf Regelverstöße aufmerksam machen oder Platzhalter für die Dokumentation einfügen.
- **Coding:** Nicht zu unterschätzen ist die Unterstützung beim Schreiben von Code. Entwicklungsumgebungen bringen eine Syntaxhervorhebung mit, durch die fehlerhafte Befehle oder Klammern schneller sichtbar werden. Ebenso weisen sie darauf hin, wenn verwendete Objekte noch nicht definiert sind oder definierte Objekte nicht verwendet werden. Daneben vervollständigen sie nach dem Eintippen der ersten Buchstaben automatisch Befehle, zeigen die Hilfe zu einer Funktion an oder fügen automatisch Grundgerüste für Funktionen und Kontrollstrukturen ein.
- **Debugging:** Unter Debugging wird die Fehlersuche in Programmcode verstanden. Einige Fehler sind so knifflig, dass sie nur schwer aufzuspüren sind. Mit einem Debugger kann man den Code Zeile für Zeile abarbeiten und dabei zusehen, wie sich die Daten (bzw. Variablen und Objekte) verändern. So werden die einzelnen Programmierschritte kontrolliert und diejenigen Stellen enttarnt, an denen sich Fehler eingeschlichen haben. Gerade in komplexen Programm-

Korrekte Verwendung

```
import os
import sys

if foo is not None:
    print(foo)

i = i + 1

if greeting:
    print("Hello")
```

Verletzung des Coding Standards

```
import sys, os

if not foo is None:
    print( foo )

i=i+1

if greeting == True:
    print("Hello")
```

Abb. 6.3 Beispiele für den Coding Standard PEP 8. Die jeweiligen Varianten funktionieren beide. Dennoch ist es für die bessere Lesbarkeit empfehlenswert, sich an einen Standard zu halten. (Quelle: van Rossum et al. (2013; https://peps.python.org/pep-0008/))

abläufen ist es hilfreich, einen Breakpoint zu setzen, der den Code entweder an einer bestimmten Zeile oder wenn eine Bedingung eingetreten ist – eine Variable wurde zum Beispiel verändert – pausiert und den Debugger an ebendieser Stelle startet.

- **Profiling:** Auch wenn Computer viele schematische Aufgaben schneller erledigen als Menschen, kann dies immer noch zu langsam sein. Vergleicht man etwa eine Million Texte miteinander, dann ergeben sich daraus bereits eine Billion Vergleiche. Selbst wenn ein einzelner Vergleich nur eine Millisekunde benötigen würde, müsste man über dreißig Jahre auf das Ergebnis warten. Auch der Arbeitsspeicher wird dabei knapp. Deshalb kann es wichtig werden, die verwendeten Algorithmen zu optimieren. Beim Profiling lässt man das Programm laufen und kann hinterher für jede einzelne Teilfunktion die Laufzeit und den Arbeitsspeicherverbrauch analysieren, um so Flaschenhälse zu identifizieren (Abb. 6.4).

Flame Graph | Data | Options

Code	File	Memory (MB)		Time (ms)
▼ runWorld	<expr>	-7257.0	7303.2	321990
▼ do.call	worldfunctions.R	-7257.0	7302.9	321970
▼ mediateMessages		-5192.8	4614.4	281120
▼ %>%	worldfunctions.R	-5188.5	4612.6	281050
▼ withVisible		-5188.5	4601.1	280900
▼ eval		-5188.5	4601.1	280900
▼ eval		-5188.5	4601.1	280900
▼ _fseq		-5188.5	4601.1	280900
▼ freduce		-5188.5	4601.1	280900
▼ function_list[[i]]		-5095.3	4415.7	278400
▶ pairwise_similarity		-5046.1	4334.0	277460
▶ addDistance		-4.3	27.6	250
▶ select		-4.3	13.5	190
▶ rename		-19.3	14.3	130
▶ filter		-21.2	8.2	130
▶ inner_join		0	4.0	80
▶ group_by				
▶ semi_join				
▶ top_n				
▶ mutate				
▶ withVisible				
▶ split_chain				
▶ lapply		0	6.8	60
▶ compiler:::tryCmpfun		-4.3	1.9	70
▶ generateLogs		-886.4	1267.1	16320

Die Funktion zum paarweisen Vergleich aller Datensätze hat in diesem Beispiel die meisten Ressourcen verbraucht: 4 GB Arbeitsspeicher und fast 5 Minuten Laufzeit.

Abb. 6.4 Profiling eines Skripts. (Quelle: eigene Darstellung)

- **Testing:** Auch wenn automatisierte Verfahren insofern reliabel sind, dass sie bei gleichem Input das immer gleiche Ergebnis ausgeben, bedeutet dies nicht, dass ein Verfahren auch wirklich valide ist und ausgibt, was es soll. Die Beurteilung darüber liegt bei den Programmierer:innen, gerade bei komplexen Operationen ist die Funktionsweise aber nicht immer gut überschaubar. Um die Qualität des Codes zu sichern und Fehler frühzeitig zu erkennen (vor allem, wenn man umfangreiche Refaktorierungen vornimmt), schreibt man idealerweise zu jeder Funktion mindestens eine zugehörige Testfunktion. Diese führt die Funktion aus, vergleicht sie mit dem zu erwartenden Ergebnis und schlägt Alarm, wenn das Resultat nicht den Erwartungen entspricht. Bei der sogenannten testgetriebenen Entwicklung (engl. *test-driven development*) werden die Tests vor der Umsetzung der eigentlichen Funktion angelegt. Gerade bei komplexen Berechnungen, etwa der Bestimmung der Ähnlichkeit zwischen zwei Texten, wird vorab der zu erwartende Wert festgelegt und erst dann die konkrete Funktion implementiert. Die Implementierung ist erfolgreich, wenn sie auch tatsächlich das festgelegte Ergebnis ausgibt. Entwicklungsumgebungen helfen bei der Entwicklung und beim Ausführen von Tests.
- **Server-Management:** Forschungssoftware umfasst auf der einen Seite kleinere Skripte für die explorative Datenanalyse, die ausschließlich auf dem eigenen Computer ausgeführt werden. Auf der anderen Seite finden sich größere Programme, die auf einem Webserver laufen oder für die Nutzung durch andere Wissenschaftler:innen veröffentlicht werden. Entwicklungsumgebungen unterstützen das Management virtueller Maschinen und Container und das Deployment, das heißt das Ausspielen auf einen Server, zum Beispiel zur Veröffentlichung einer Software (siehe Abschn. 6.3).

Nicht alle diese Funktionen werden bei der Arbeit mit Integrierten Entwicklungsumgebungen von Anfang an benötigt, es lohnt sich aber bei der Wahl der Entwicklungswerkzeuge darauf zu achten, welche Optionen perspektivisch zur Verfügung stehen. Das Angebot bestimmt gewissermaßen die Nachfrage – nach einiger Zeit der Eingewöhnung lassen sich immer mehr hilfreiche Features entdecken. Zum Erlernen von Computational Methods gehört dazu, sich nach und nach Entwicklungsroutinen anzueignen, die man später nicht mehr missen möchte. Zu Beginn sind die grafischen Oberflächen vor allem hilfreich, um den Überblick über das Projekt zu wahren und weil sie viele Hilfestellungen anbieten. Um die möglichen Optionen zu erkunden, kann man sich durch die Menüs klicken, statt in Dokumentationen zu lesen. Achten Sie dennoch von vornherein darauf, sich die Bedienung über die Tastatur anzueignen und halten Sie Ausschau nach Shortcuts. Während Computational Methods die Datenanalyse automatisieren, kann eine Ent-

wicklungsumgebung die Entwicklungsarbeit automatisieren. Spätestens wenn man sich im Klickverhalten wiederholt – weil man zum Beispiel immer wieder auf RUN klickt, um ein Skript laufen zu lassen – sollte man auf das dazugehörige Shortcut umsteigen. Auch auf die Shortcuts zum Kommentieren von Text (häufig Strg + / oder Command + /), die Autovervollständigung (Tabulatortaste sowie Strg + Leertaste bzw. Command + Leertaste) und die Hilfe (F1) wollen Sie nach kurzer Zeit nicht mehr verzichten. Im Zeitverlauf tritt die Entwicklungsumgebung dadurch immer mehr in den Hintergrund und die Produktivität nimmt zu. Sie fangen an, die Gedanken in Code umzusetzen und müssen nicht mehr über das „Wie?", sondern nur noch über das „Was?" nachdenken.

Einführend wurde bereits darauf hingewiesen, dass sich Entwicklungsumgebungen dahingehend unterscheiden, ob sie auf eine bestimmte Programmiersprache abgestimmt oder ob sie universell einsetzbar sind. Eine weitere Unterscheidung betrifft den Entwicklungsprozess. Bei der interaktiven Entwicklung werden Codezeilen oder kurze Schnipsel direkt ausgeführt, um das Ergebnis zu betrachten. Für wissenschaftliche Zwecke ist dies nützlich, wenn explorative Datenanalyse betrieben wird bzw. ein Skript nach und nach aufgebaut und nur für einmalige Analysen verwendet werden soll. Insbesondere Notebooks, die in JupyterLab für die Programmiersprachen Python, R und Julia eingesetzt werden, fördern diesen Entwicklungsstil und dokumentieren gleichzeitig die Ergebnisse. Bei der nichtinteraktiven Entwicklung werden dagegen erst Funktionen und Klassen (siehe zur objektorientierten Programmierung Kap. 5) aufgebaut, um dann das ganze Programm ablaufen zu lassen. Hier steht die effiziente, elegante und erweiterbare Konstruktion des Programms im Vordergrund. Dieser Stil ist angebracht, sobald ein Programm längerfristig genutzt werden soll – zum Beispiel, weil es eine grafische Benutzeroberfläche hat oder in Datenanalyseprozesse von Organisationen eingebettet ist. Für die nichtinteraktive Entwicklung geeignet sind etwa PyCharm oder Visual Studio Code.

Schließlich sollte man bei der Wahl der Entwicklungswerkzeuge darauf achten, inwiefern die verarbeiteten Daten einsehbar sind. In RStudio sind die aktuell erzeugten Objekte und Datensätze jederzeit gut sichtbar, während sie in JupyterLab erst über `print()`-Befehle ausgegeben werden müssen oder bei der nichtinteraktiven Entwicklung nur mit dem Debugger (über Breakpoints) zugänglich sind.

Beachten Sie schließlich, dass sich Entwicklungswerkzeuge genauso weiterentwickeln wie die Programmiersprachen und die Verfahren automatisierter Datenerhebung und -analyse. Deshalb haben die genannten Empfehlungen möglicherweise eine geringe Halbwertzeit – erkunden Sie selbst, welche Entwicklungsumgebung für Sie und das Team am besten geeignet ist.

Übungsfragen

1. Was versteht man unter der Refaktorierung von Code?
2. Suchen Sie sich ein selbstgeschriebenes Python-Skript und prüfen Sie, ob es dem Codingstandard PEP 8 entspricht!
3. Finden Sie in einer Entwicklungsumgebung Ihrer Wahl heraus, mit welchem Shortcut Kommentare angelegt werden können und mit welcher Taste ein Befehl automatisch vervollständigt wird!
4. Probieren Sie für eine Programmiersprache Ihrer Wahl das Debugging in einer integrierten Entwicklungsumgebung aus: Setzen Sie einen Breakpoint in einem Skript, starten Sie das Skript und inspizieren Sie die Variablen, während Sie den Code schrittweise ablaufen lassen!

6.3 Laufzeitumgebungen und virtuelle Boxen

Die Welt der Computer ist vielfältig: In den verschiedenen Geräten sind je andere Prozessoren, Festplatten, Schnittstellen, Grafikkarten und Zubehöre verbaut oder Betriebssysteme installiert. Durch die stetige Weiterentwicklung der Technik sind Computersysteme zudem unbeständig und kurzlebig. Nach einigen Jahren in Gebrauch sind Computer meist schon nicht mehr auf dem aktuellen technischen Stand und werden durch Nachfolgemodelle ersetzt. Die Vielfalt der Computersysteme fördert Innovation, geht aber auch mit einigen Herausforderungen einher. Diesen begegnet man spätestens, sobald man selbst Programme entwickelt, die nicht nur auf dem eigenen Computer laufen sollen. Dies ist beispielsweise der Fall, wenn man ein R- oder Python-Skript im wissenschaftlichen Kontext für andere Wissenschaftler:innen bereitstellen will und sichergehen möchte, dass die eigene Arbeit auch nach Jahren noch reproduzierbar ist – oder weil man eine Forschungssoftware entwickelt, die auf möglichst vielen verschiedenen lokalen oder serverbasierten Systemen gleichermaßen laufen soll.

Um in der vielfältigen und dynamischen Welt der Computersysteme Stabilität zu garantieren, müssen die Laufzeitbedingungen fixiert werden. Eine erste Lösung dafür besteht darin, in Skripten die konkreten Packageversionen anzugeben, sodass man sich später mit einem Paketmanager die gleichen Komponenten noch einmal installieren kann. Unter Python wird dazu etwa der Paketmanager pip verwendet und zusätzlich lassen sich in Python sogenannte virtuelle Umgebungen (engl. *virtual environment*, kurz *venv*) einrichten, in denen die für ein Skript benötigten Packages unabhängig von anderen Skripten installiert werden.

Mitunter hängt eine Software aber vom Gesamtsystem ab – etwa, wenn man einen bestimmten Datenbankserver benötigt oder eine Komponente nur für ein spezielles Betriebssystem verfügbar ist. Statt alle benötigten Komponenten einzeln auf einem Computer zu installieren, können in diesem Fall virtuelle Laufzeitumgebungen definiert werden, die von dem eigenen Computersystem losgelöst und auf andere Computer übertragbar sind. Indem ein virtueller Computer im eigenen Computer eingerichtet wird, ergibt sich ein weiterer Vorteil: Man kann sich das Beste aus allen Welten zusammenstellen. So lassen sich etwa auch Linuxprogramme auf Windowscomputern verwenden.[8] Insbesondere Serveranwendungen wie Datenbankmanagementsysteme oder Webseiten (siehe Kap. 3) sind häufig für Linux entwickelt, sodass sich mit der Virtualisierung zum Beispiel gut SQL-Datenbankserver auf dem eigenen Computer einrichten lassen.

Bei der Umsetzung von Virtualisierung spielen in der Regel vier Schichten zusammen:

- Das **Wirtssystem** (Host) ist der eigene Computer, beispielsweise ein PC mit einem Intel-Prozessor, 16GB Arbeitsspeicher und mit einem Betriebssystem wie Windows oder macOS.
- Auf dem Host läuft ein sogenannter **Hypervisor**. Mit diesem kann eine virtuelle Maschine erzeugt werden, die einen selbst gewählten Prozessor, Arbeitsspeicher, Festplatten oder auch Bildschirme simuliert.
- In der virtuellen Maschine wird ein **Gastsystem** installiert, das auf die Ressourcen des Wirtssystems zugreift. Typischerweise sind das Linuxsysteme wie Ubuntu oder Debian.
- Zur Einrichtung des Gastsystems wird mitunter Provisionierungssoftware wie Ansible[9] verwendet. Die **Provisionierung** stellt sicher, dass die benötigten Datenbanken eingerichtet werden und Programme und Packages bei allen Nutzenden mit den gleichen Einstellungen vorhanden sind.

Dieser Aufbau führt dazu, dass erstens die Umgebung für eine bestimmte Forschungssoftware von der Umgebung des eigenen Computers isoliert ist und zweitens die identische Umgebung auf verschiedenen Computern hergestellt werden kann. Dabei kommen aktuell vor allem zwei Techniken (oder ähnliche Alternativen) zum Einsatz, Vagrant und Docker. Während mit Vagrant vollständig auto-

---

[8] Windows unterstützt mit dem Windows-Subsystem for Linux (WSL) seit einigen Jahren die Möglichkeit, auch ohne zusätzliche Virtualisierungssoftware direkt Linux-Programme auszuführen, siehe Microsoft (2022c; https://docs.microsoft.com/de-de/windows/wsl/).

[9] Siehe AnsibleWorks (2022; https://www.ansible.com/).

nome virtuelle Maschinen eingerichtet werden, erzeugt Docker sogenannte Container. Ein einzelner Container enthält lediglich die konkret benötigten Komponenten, zum Beispiel einen Datenbankserver. Eine vollständige Laufzeitumgebung setzt sich in der Regel aus mehreren spezialisierten Containern zusammen. Hinzu kommt, dass Container stärker in das Wirtssystem eingebettet sind und dadurch schneller starten.

Ein Anwendungsfall für die Virtualisierung mit Vagrant oder Docker besteht darin, einen vollständigen Webserver auf dem eigenen Computer einzurichten. Das ist nicht nur eine Grundlage für die Entwicklung von Webseiten – wenn etwa im wissenschaftlichen Kontext Tools zur Datenaufbereitung im Web bereitgestellt oder geisteswissenschaftliche Editionen historischer Schriften veröffentlicht werden. Virtuelle Boxen lassen sich auch dazu einsetzen, um beispielsweise eine Kopie der Wikipedia inklusive der dazugehörigen Datenbank auf einem Computer zu erstellen, sodass man diese Daten ohne Webscraping oder APIs analysieren kann.[10]

6.3.1 Der LAMP-Stack

Um einen vollständigen Webserver aufzusetzen, ist eine Reihe an Komponenten nötig. Die in den folgenden Abschnitten vorgestellten Setups sind typischerweise im Kontext der Webentwicklung anzutreffen, aber gleichzeitig auch eine solide Ausgangsbasis für viele weitere Computational-Methods-Szenarien. Eine klassische Webseite baut auf dem sogenannten LAMP-Stack auf, die Abkürzung setzt sich aus vier verbreiteten Open-Source-Anwendungen zusammen (Kunze 1998):

- **Linux**: Webserver sind ganz normale Computer, die aber häufig ohne Bildschirm und Tastatur auskommen und nicht Windows oder MacOS, sondern verschiedene Varianten von Linux als Betriebssystem verwenden. Für Entwicklungssysteme werden beispielsweise Ubuntu Server[11] und Alpine[12] eingesetzt.

---

[10] Zudem ist ein Grundverständnis der Infrastruktur hinter einer Webseite hilfreich für die automatisierte Datenerhebung im Web (siehe Kap. 7). Zu einer sozialwissenschaftlichen Perspektive auf die Infrastruktur des Web siehe unter anderem van Dijck (2020).

[11] Siehe Canonical (2022; https://ubuntu.com/).

[12] Siehe Alpine Linux Development Team (2022, https://alpinelinux.org/).

- **Apache**: Für die Auslieferung der Webseiten über das Internet wird eine Webserversoftware benötigt. Das Apache-Projekt[13] hat in diesem Bereich eine lange Tradition: Die dahinterstehende, ehrenamtlich organisierte Apache Software Foundation ist eine der bedeutendsten Organisationen im Bereich der Open-Source-Software.
- **MySQL**: Die Daten, aus denen eine Webseite generiert wird, können unter anderem in einer relationalen SQL-Datenbank verwaltet werden (siehe Abschn. 3.6). MySQL oder alternativ die kompatible Open-Source-Software MariaDB sind dafür eine klassische Wahl.[14]
- **PHP**: Selbst wenn Sprachen wie Python und JavaScript durchaus in der Webentwicklung beliebt sind und der Tod von PHP[15] immer wieder vorhergesagt wird, ist diese Sprache nach wie vor die vorherrschende Programmiersprache für Webseiten.[16]

Prinzipiell lassen sich all diese Komponenten einzeln installieren. Um jedoch das Zusammenspiel zu gewährleisten, ist ein recht hoher Konfigurationsaufwand nötig. Vagrant und Docker helfen dabei, den eigenen Computer mit wenig Aufwand in einen vollwertigen Server zu verwandeln. Alle Dateien und Funktionen bleiben dennoch lokal und sind nicht über das Internet erreichbar.

6.3.2 Vagrant

Für die Koordination des Zusammenspiels von Wirtssystem, Hypervisor und Gastsystem finden sich verschiedene kommerzielle und nichtkommerzielle Programme. Eigenständige virtuelle Maschinen lassen sich beispielsweise mit der Open-Source-Software VirtualBox[17] erzeugen, für die Einrichtung einer solchen Box ist zudem Vagrant[18] empfehlenswert. Nach der Installation der beiden Komponenten kann mit nur einem einzigen Befehl auf der Kommandozeile eine vollständige Serverumgebung erzeugt werden. Dazu erstellt man mit einem Texteditor eine

[13] Siehe Apache Software Foundation (2022b; https://httpd.apache.org/).

[14] Siehe MariaDB Foundation (2022; https://mariadb.org/).

[15] Siehe The PHP Group (2022; https://www.php.net/).

[16] In einer Untersuchung von W3Techs (2022) benutzten 78 % der Seiten PHP und in einer Befragung von JetBrains (2022b) nutzten 85 % der Webentwickler:innen PHP – auch wenn es sich nicht um repräsentative Erhebungen handelt, vermitteln die Zahlen einen Eindruck.

[17] Siehe Oracle (2022c, https://www.virtualbox.org).

[18] Siehe HashiCorp (2022a, https://www.vagrantup.com/)

Konfigurationsdatei mit dem Namen *Vagrantfile*. Diese Datei kann anschließend an andere Entwickler:innen weitergegeben werden (☛ *Repositorium*).

Um die einzelnen Bestandteile des Vagrantfile zu verstehen, ist die Dokumentation[19] hilfreich. Im Vagrantfile wird zunächst festgelegt, welche Box verwendet werden soll. Eine Box enthält das benötigte Betriebssystem, also das Gastsystem, sie wird häufig von anderen Anbietern heruntergeladen, statt dass sie selbst erstellt wird. Im folgenden Beispiel wird *scotch/box*[20] verwendet. Diese vorbereitete Box enthält ein Ubuntu-System mit allen Komponenten, die für die Webentwicklung benötigt werden, insbesondere einen PHP-Webserver und einen mySQL-Server:[21]

```
Vagrant.configure("2") do |config|
    config.vm.box = "scotch/box"
    config.vm.network "private_network", ↵
        ip: "192.168.33.10"
    config.vm.hostname = "scotchbox"
    config.vm.synced_folder ".", "/var/www ",↵
        :mount_options =>; ["dmode=777", "fmode=666"]
end
```

Wenn Sie Vagrant installiert und eine Vagrant-Datei erstellt haben, starten und initialisieren Sie die Box ausgehend von dem Ordner mit dem Vagrantfile mit nur einem Befehl auf der Kommandozeile (siehe Abschn. 1.2.2):

```
vagrant up
```

Damit Skripte und Daten zwischen dem Wirts- und dem Gastsystem ausgetauscht werden können, werden Ordner zwischen beiden Systemen synchronisiert (`synced_folder`). Im Beispiel wird durch die Angabe des Punktes der Ordner, in dem das Vagrantfile auf dem Wirtssystem liegt, mit dem Ordner */var/www* innerhalb des Gastsystems synchronisiert. Legt man dort ein Skript ab – die Entwicklung kann auf dem Wirtssystem erfolgen –, so kann es über den Webbrowser aufgerufen werden. Dazu gibt man in die Adressleiste des Browsers die im

---

[19] Siehe HashiCorp (2022b, https://www.vagrantup.com/docs).

[20] Siehe Digital Ocean (2022; https://github.com/scotch-io/scotch-box).

[21] Das Umbruchszeichen ↵ wurde lediglich aus Darstellungsgründen eingefügt, die jeweiligen Zeilen müssen ohne Zeilenumbruch eingegeben werden.

Vagrantfile vorgegebene IP-Adresse http://192.168.33.10 ein.[22] Die Box ist so weit vorkonfiguriert, dass sich in der Box beispielsweise Content-Management-Systeme wie WordPress nutzen lassen. Auf der Webseite von ScotchBox und im ☛ *Repositorium* des Buchs finden Sie eine kurze Anleitung zur Einrichtung der Box mit einer Miniwebseite.

6.3.3 Docker

Eine Alternative zur Einrichtung von vollständigen virtuellen Maschinen bietet Docker.[23] Die Software kann auf der Download-Seite von Docker[24] heruntergeladen und installiert werden.[25] Unter Windows wird zusätzlich das Windows Subsystem for Linux (WSL) benötigt.[26] Auch hier gilt: man muss ein wenig Erfahrung sammeln, um die vielen Begriffe aus der Welt der DevOps – mit diesem Begriff wird das Zusammenspiel von Softwareentwicklung und IT-Infrastruktur bezeichnet – zu verstehen.

Zentral bei der Arbeit mit Docker sind die sogenannten Container. Docker-Container verhalten sich wie virtuelle Computer, stellen anders als bei der Arbeit mit Vagrant aber immer nur einzelne Dienste zur Verfügung, beispielsweise in einem Container einen Webserver oder in einem anderen Container einen Datenbankserver. Diese Container bauen auf Images mit Betriebssystemen wie Ubuntu auf und fügen die für einen Dienst nötigen Schichten und Pakete hinzu. Ein fertiger Container kann wiederum als Image abgespeichert und an andere Entwickler:innen weitergegeben werden. Das Image kann sowohl direkt zum Starten von Containern verwendet werden, als auch um wiederum andere Images mit weiteren Funktionen abzuleiten. Fertige Images werden über sogenannte Registries verwaltet, beispielsweise in der offiziellen Docker-Registry.[27]

---

[22] Wird keine IP-Adresse konfiguriert, so ist die Box unter den Standardadressen für lokale Server erreichbar: http://localhost und http://127.0.0.1.

[23] Indem zwei der oben benannten Schichten zusammengeführt werden, starten Docker-Systeme schneller: Der Hypervisor und das Gast-Betriebssystem werden durch die Docker-Engine ersetzt.

[24] Siehe Docker (2022a; https://www.docker.com/products/docker-desktop).

[25] Achten Sie unter Windows darauf, bei Nachfragen während der Installation die Linux-Container zu aktivieren.

[26] Siehe Microsoft (2022c; https://docs.microsoft.com/en-us/windows/wsl/install).

[27] Siehe Docker (2022b; https://hub.docker.com/).

Ein Docker-Container wird durch eine Datei mit dem Namen *Dockerfile* definiert, sie kann einfach mit einem Texteditor angelegt werden (☛ *Repositorium*). Diese Datei beginnt mit der Benennung eines Basisimages (`FROM`) und der Zuweisung eines Arbeitsverzeichnisses innerhalb des Containers (`WORKDIR`). Anschließend werden Befehle angegeben, die bei der Ersteinrichtung des Containers abgearbeitet werden sollen (`RUN`). Die im folgenden gekürzten Beispiel verwendeten Linux-Befehle[28] installieren nach der Aktualisierung des Paketmanagers apt-get einen SQL-Client, damit auf den Datenbankserver zugegriffen werden kann:

```
FROM php:7.1.2-apache
WORKDIR /var/www/html

# Add SQL client
RUN apt-get update
RUN apt-get install -y mysql-client
RUN rm -rf /var/lib/apt
RUN docker-php-ext-install pdo_mysql
```

Einzelne über ein Dockerfile definierte Container werden mit dem Befehl `docker up` gestartet. Eine vollständige Entwicklungsumgebung benötigt jedoch in der Regel mehrere Dienste, die über unterschiedliche Container bereitgestellt werden. Für die klassische Webentwicklung benötigt man mindestens einen Webserver und einen Datenbankserver. Damit man nicht jeden Container einzeln starten und konfigurieren muss, kann man das Zusammenspiel mit Docker Compose orchestrieren. Dazu wird eine Datei *docker-compose.yml* angelegt und darin festgelegt, welche Container aus welchen Images gestartet werden sollen. Im folgenden Beispiel werden zwei Container verwendet: ein PHP-Webserver `server_php` und ein SQL-Datenbankserver `server_sql`. Ein Unterschied zwischen den beiden Containern besteht darin, dass für `server_php` durch die `build`-Anweidung ein Dockerfile im Unterordner *php* verwendet wird, während der SQL-Dienst direkt das MariaDB-Image aus der Docker-Registry verwendet:

[28] Bei diesen Befehlen handelt es sich einerseits um Linux-Grundbefehle (`rm, apt-get`) und andererseits um spezifische Docker-Befehle (`docker-php-ext-install`). Wenn man neu in dieser Welt ist, empfiehlt es sich zunächst, Dockerfiles zu recherchieren, von anderen Nutzenden zu übernehmen und sich die Befehle nach und nach anzueignen. Der Aufbau von Dockerfiles ist in der Docker-Dokumentation beschrieben: Siehe Docker (2022c; https://docs.docker.com/reference/).

```
services:
    php:
        build: ./php/
        container_name: "server_php"
        ports:
            - "80:80"
            - "443:443"
        volumes:
            - "./html:/var/www/html:rw"
        networks:
            - database
        depends_on:
            - sql
        environment:
            MYSQL_ROOT_PASSWORD: "root"
    sql:
        image: mariadb:10.3
        container_name: "server_sql"
        ports:
            - "3306:3306"
        networks:
            - database
        environment:
            MYSQL_ROOT_PASSWORD: "root"
            MYSQL_DATABASE: "devel"
networks:
    database:
        driver: bridge
```

Die beiden Container hängen miteinander zusammen: Der Webserver wird nur gestartet, wenn auch der Datenbankserver vorhanden ist (`depends_on`-Anweisung). Die beiden Container tauschen sich dann über ein Docker-internes Netzwerk (`networks`-Anweisung) mit dem Namen „database" untereinander aus. Während im Datenbankcontainer eine Datenbank mit dem Namen „devel" erzeugt wird, kann vom PHP-Container mit dem Benutzernamen „root" und dem Passwort „root" auf diese Datenbank zugegriffen werden (`environment`-Anweisung) – deshalb wurde im oben besprochenen Dockerfile der SQL-Client installiert.

Bislang ist eine in sich geschlossene Containerwelt beschrieben worden – wie kann vom Wirtssystem auf diese Server zugegriffen werden? Hier sind zwei Wege zu unterscheiden, einerseits der Netzwerkzugriff und andererseits der Zugriff auf Dateien. Ausgehend vom Wirtssystem gelingt der Netzwerkzugriff auf den Webserver über den Browser mit der Adresse http://localhost. Dazu wurde in der Konfigurationsdatei eine Weiterleitung von Port 80 des Wirtssystems auf den Port 80 des Gastsystems festgelegt (`ports`-Anweisung). Ports sind Türen im Netzwerk, durch die Daten zwischen verschiedenen Computern übertragen werden, seien es physische oder virtuelle Computer. Der Port 80 wird im Internet dazu verwendet, Webserver bereitzustellen. Der SQL-Server stellt seine Dienste dagegen unter dem Port 3306 zur Verfügung. Jeder Server kann Ports mit nahezu beliebigen Nummern öffnen: Auf einem Server ist immer ein bestimmter Dienst mit einem einzelnen Port verbunden. Die im Beispiel verwendeten Nummern sind die Standardports für Web- und Datenbankserver. Im hier verfolgten Szenario sind diese Ports aber nur lokal auf dem eigenen Computer erreichbar, dafür sorgt unter anderem die Firewall des eigenen Computers. Für den Dateiaustausch zwischen den Systemen wird das Unterverzeichnis *html* im Wirtssystem mit dem Verzeichnis */var/www/html* im Gastsystem synchronisiert (`volumes`-Anweisung). Dort können Skripte abgelegt werden, die vom PHP-Server verarbeitet werden. Der Webserver sorgt dann in der Standardkonfiguration dafür, dass beim Aufrufen von http://localhost im Browser das Skript *html/index.php* auf dem Server ausgeführt wird.

Ein Dockersystem besteht also aus Images, von denen ausgehend Container gestartet werden, die ihre Dienste in Netzwerken bereitstellen und deren Dateien über synchronisierte Ordner erreichbar sind. Wie bei der Arbeit mit Vagrant reicht ein einziger Befehl auf der Kommandozeile (siehe Abschn. 1.2.2). Um das so definierte Ensemble von Containern zum Leben zu erwecken, muss man sich im Ordner der *docker-compose.yaml* befinden und die Kommandozeile ggf. im Administratormodus starten:

```
docker compose up -d
```

Die benötigten Images werden automatisch heruntergeladen bzw. erstellt. Anschließend werden die Container gestartet, mit dem Dateisystem verbunden und miteinander vernetzt. Die Option `-d` sorgt dafür, dass die Kommandozeile nach dem Starten wieder für weitere Befehle zur Verfügung steht (detached mode). Das kann beim ersten Mal eine Weile dauern, die Zeit reicht für einen Tee oder Kaffee. Schon beim zweiten Mal sollten die Container aber blitzschnell starten – das ist ein Vorteil gegenüber der vollständigen Virtualisierung, wie sie etwa mit Vagrant möglich wäre. Für die weitere Arbeit mit Docker gibt es eine Vielzahl nützlicher Be-

fehle. Um die weiteren Möglichkeiten besser kennenzulernen, empfiehlt sich ein Blick in die Dokumentation von Docker.[29]

6.3.4 Anwendungsbeispiel: Mit SQL-Datenbanken arbeiten

Angenommen Sie wollen die Wikipedia auf Ihrem Computer auswerten, dann können Sie sich einen vollständigen SQL-Dump herunterladen und für die Analyse auf Ihrem Computer in eine lokale Datenbank einspielen (siehe Abschn. 3.6). Ein SQL-Dump besteht aus Befehlen, um die Tabellen und Daten zu erstellen und kann deshalb auch nur in Verbindung mit einem Datenbankmanagementsystem genutzt werden. Wenn Sie entsprechend des vorangegangenen Abschnitts eine Docker-Umgebung einrichten, dann sind Sie bestens dafür ausgestattet (☛ *Repositorium*). Die folgenden Schritte sind an einem kleineren Beispiel ausgerichtet, das sich auf andere SQL-Dumps übertragen lässt – es wird ein Ausschnitt aus der International Movie Database in eine SQL-Datenbank eingespielt.

Schritt 1 – Zur Orientierung: Ein- und Auftauchen in den SQL-Server Bei der Arbeit mit Docker ist zunächst wichtig, dass man versteht, wie die Systeme ineinander verschachtelt sind. Bevor es im nächsten Schritt um das Einlesen der Datenbank geht, wird deshalb zunächst das Ein- und Auftauchen in den Server demonstriert. Öffnen Sie eine Kommandozeile im Docker-Arbeitsverzeichnis und starten Sie die Container:

```
docker compose up -d
```

Noch befinden Sie sich im Betriebssystem Ihres Computers, also etwa unter Windows oder MacOS. Von hier aus können Sie in den PHP-Container eintauchen, indem Sie dort eine Kommandozeile starten, typischerweise wird dafür auf Linuxsystemen Bash verwendet:

```
docker exec -it webdock_php /bin/bash
```

Sie tippen anschließend zwar im gleichen Fenster, sind tatsächlich aber in einen anderen (virtuellen) Computer eingetaucht. Das erkennen Sie auch daran, dass sich die Darstellung der Kommandozeile leicht verändert hat und nun mit einer Angabe wie `root@a74182ba3e1e:/var/www/html#` beginnt (Abb. 6.5). Vor dem

[29] Siehe Docker (2021; https://docs.docker.com/).

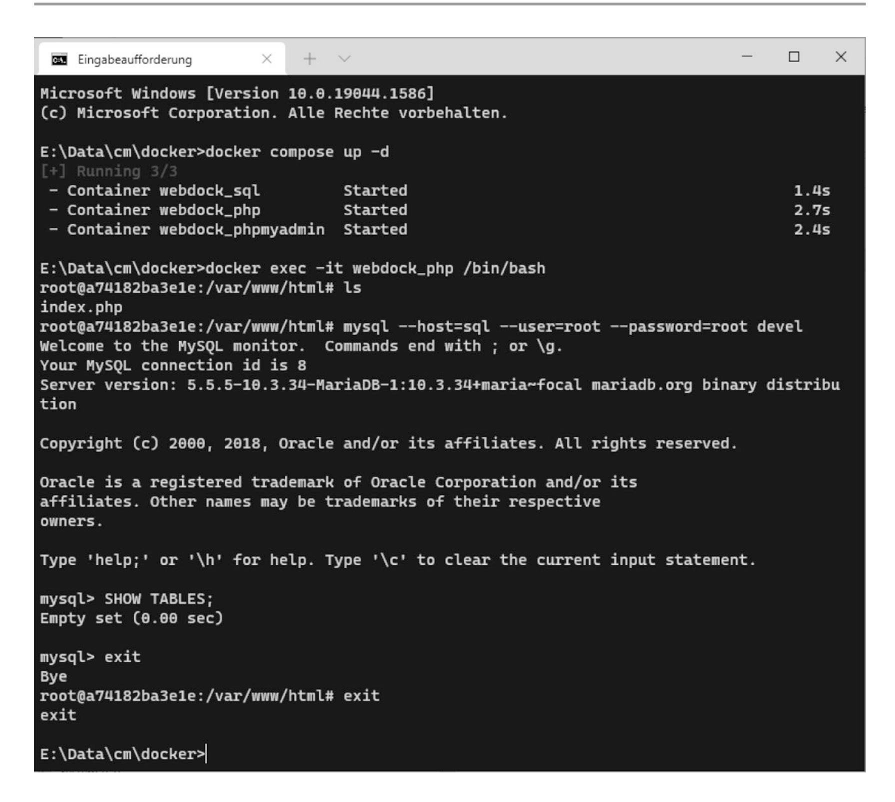


Abb. 6.5 Ein- und Auftauchen auf der Kommandozeile. (Quelle: eigene Darstellung)

`@`-Zeichen steht der Nutzername im Container, es folgen eine Nummer oder ein Name, die den Container identifizieren, und schließlich ist das aktuelle Arbeitsverzeichnis angegeben. Führen Sie den Befehl `ls` aus, um zu sehen, welche Dateien sich aus Sicht des Containers dort befinden.

Aus dem PHP-Container können Sie sich nun zum SQL-Server verbinden, auf dem durch das vorgegebene Docker-Setup bereits eine leere Datenbank „devel“ eingerichtet ist:

```
mysql --host=sql --user=root --password=root devel
```

Und wieder sind Sie in eine neue Ebene abgetaucht, die Kommandozeile ändert sich dahingehend, dass sie mit `mysql>` beginnt. Sie befinden sich nun im SQL-Server und können dort Befehle auf der Datenbank ausführen (siehe Abschn. 4.1.4):

```
SHOW TABLES;
```

Aktuell ist die Datenbank noch leer, es werden keine Tabellen angezeigt, deshalb wird im nächsten Schritt ein SQL-Dump in diese Datenbank eingespielt. Tauchen Sie zunächst wieder auf, indem Sie den SQL-Server verlassen:

```
exit
```

Um wieder zum Ausgangssystem zurückzukehren, geben Sie ein weiteres Mal „exit“ ein:

```
exit
```

Bevor Sie weitergehen, vergegenwärtigen Sie sich die verschiedenen Ebenen, auf denen Sie gerade unterwegs waren: das Wirtssystem (z. B. Windows oder Mac), das Gastsystem (Ubuntu im Docker-Container) und der SQL-Server (Abb. 6.6). Der nächste Schritt beginnt noch einmal mit dem Eintauchen in das Gastsystem.

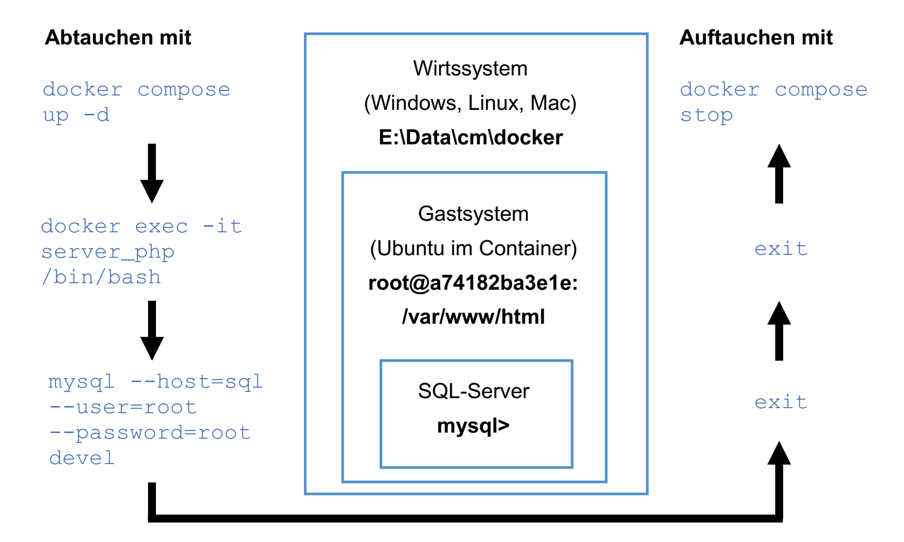


Abb. 6.6 Ein- und Auftauchen in den SQL-Server. Die Befehle sind zu Darstellungszwecken umgebrochen. In der Kommandozeile müssen sie auf einer Zeile eingegeben werden. (Quelle: eigene Darstellung)

Schritt 2 – Einspielen des Dumps Wenn Sie aus dem ☛ *Repositorium* des Buchs den Ordner zum Kapitel heruntergeladen haben, dann befindet sich im Unterordner *dump* ein gezippter SQL-Dump (siehe Abschn. 3.6). Öffnen Sie wieder die Kommandozeile, starten Sie Docker und begeben Sie sich in den PHP-Container (siehe oben). Gehen Sie mit folgenden Befehlen in das Verzeichnis mit dem SQL-Dump (`cd` = change dir) und entpacken Sie die ZIP-Datei mit `unzip`. Sie können anschließend mit `ls` kontrollieren, ob eine SQL-Datei vorliegt:

```
cd /var/dump
unzip imdb.zip
```

Statt dass Sie sich nun zum SQL-Server verbinden und Befehle eingeben, spielen Sie die SQL-Satements mit der Pipe < direkt aus der SQL-Datei in die Datenbank *devel* ein (alles muss in einer Zeile eingegeben werden):

```
mysql --host=sql --user=root --password=root devel < /var/dump/imdb.sql
```

Wenn Sie sich anschließend, wie oben beschrieben, die Tabellen in der Datenbank ausgeben lassen, sollten diese nicht mehr leer sein. Auch wenn Sie im Browser die Adresse http://localhost aufrufen, sollte ein kleiner Ausschnitt an Personen mit Geburts- und Sterbedaten sichtbar werden.

Schritt 3 – Zugriff auf die Daten Für den Zugriff auf die Daten gibt es viele unterschiedliche Optionen, von grafischen Datenbankprogrammen wie DBeaver[30] oder HeidiSQL[31] bis zur Kommandozeile. Sobald der Server läuft, brauchen Sie lediglich die Serveradresse bzw. den Hostnamen („localhost") sowie Benutzername und Passwort (beides „root"). Suchen Sie sich selbst eines der Programme aus und versuchen Sie damit die Verbindung herzustellen.

Für die Datenanalyse können Sie zudem mit Skripten auf die Datenbank zugreifen. Unter R gelingt das etwa mit dem Package *RMySQL* (Ooms et al. 2021). Im Zusammenspiel mit *dbplyr* (Wickham, Girlich & Ruiz 2022; nicht zu verwechseln mit *dplyr*) können Sie für den Datenabruf auf Tidyverse-kompatible Befehle zurückgreifen, statt SQL-Statements zu formulieren. Zunächst stellen Sie dafür die Verbindung zur Datenbank her:

[30] Siehe DBeaver (2022; https://dbeaver.io/).

[31] Siehe Becker (2022; https://www.heidisql.com/).

```
# Packages
library(tidyverse)
library(RMySQL)
library(dbplyr)

# Verbindung herstellen
con <- dbConnect(
  RMySQL::MySQL(),
  host = "localhost",
  port = 3306,
  username = "root",
  password = "root",
  dbname = "devel"
)
```

Das daraus hervorgehende Verbindungsobjekt `con` kann dann mit der `tbl()`-Funktion zur Auswahl einer Tabelle innerhalb der Datenbank verwendet werden. Mit `collect()` wird der Inhalt der Tabelle in einem Dataframe für die weitere Analyse abgelegt:

```
people <- tbl(con, "people")
people <- collect(people)
count(people, born, sort = T)
```

Lassen Sie die Zeile mit dem `collect()`-Befehl einmal weg, die Auszählung sollte dennoch klappen. Denn es ist nicht unbedingt nötig, die Daten vollständig zu laden – wenn Sie Befehle wie `filter()`, `select()` oder `count()` verwenden, kann *dbplyr* daraus im Hintergrund passende SQL-Abfragen formulieren, die dann auf dem Server ausgeführt werden, ohne alle Daten zu R zu transferieren. Die Daten werden dann automatisch am Ende eingesammelt. Das Package versucht also vor einem `collect()`-Befehl die Datenaufbereitung auf dem Server durchzuführen, hinter diesem Befehl findet die Auswertung dagegen in R statt.

Mit Python ist die Vorgehensweise ganz ähnlich, wenn Sie die Packages *sqlalchemy* (Bayer 2012) und *pymysql* (Matsubara et al. 2016) in Kombination mit *pandas* (siehe Abschn. 5.2) verwenden. Zunächst wird eine Verbindung hergestellt:

```
# Packages
from sqlalchemy import create_engine
import pymysql
import pandas as pd
```

```
# Verbindung herstellen
db_str = 'mysql+pymysql://root:root@localhost/devel'
db_con = create_engine(db_str)
```

Die Verbindungsdaten werden hier über einen Verbindungsstring angegeben. Dieser enthält eine Protokollangabe `mysql+pymysql`, dann vor dem Doppelpunkt den Nutzernamen `root` und danach das Passwort `root` sowie nach dem `@`-Zeichen den Hostnamen `localhost` und die gewünschte Datenbank `devel`. Über das Verbindungsobjekt `db_con` lassen sich nun SQL-Befehle absetzen, um das Ergebnis in einem Dataframe abzulegen:

```
df = pd.read_sql ('SELECT * FROM people', con = db_con)
display(df)
```

Sie können sich damit beispielsweise an einer Visualisierung der Geburts- und Sterbejahre im Datensatz versuchen. Vergessen Sie nicht, die Datenbankverbindung abschließend mit `db_con.dispose()` wieder zu beenden!

Übungsfragen

1. Wozu wird die Virtualisierung von Maschinen eingesetzt?
2. Was ist ein LAMP-Stack und wofür wird er eingesetzt? Recherchieren Sie, was die Stacks LEMP, MEAN und XAMPP enthalten!
3. Was unterscheidet Vagrant von Docker?
4. Wie starten Sie Docker-Container?
5. Unter welcher Adresse können Sie mit einem Browser auf einen lokalen Webserver zugreifen?
6. Richten Sie die im ☛ *Repositorium* des Buchs definierte Vagrant-Box oder die dort vorgegebenen Docker-Container ein!
7. Was ist ein SQL-Dump und wie gehen Sie vor, um damit zu arbeiten?
8. Joinen Sie die Tabellen „people“ und „crew“ im IMDb-Datensatz (☛ *Repositorium*), um die Anzahl der Schauspieler:innen, Regisseur:innen und Produzent:innen von Filmen nach Jahren auszuzählen!
9. Welche Möglichkeiten kennen Sie, um auf SQL-Datenbanken zuzugreifen?

6.4 Parallelisierung mit Cores, Clustern und Cloud Computing

Die Zeit eines Menschen ist begrenzt. Und auch ein einzelner Computer schafft es bei einigen Aufgaben nicht, in überschaubarer Zeit zu einer Lösung zu kommen. Ein Grund dafür kann sein, dass eine große Datenmenge zu bearbeiten ist, eine Funktion also zum Beispiel auf Millionen oder Milliarden von Fällen angewendet werden soll. Doch auch kleine Datensätze können je nach Fragestellung schnell zu praktisch unendlichem Zeitaufwand führen. Sollen etwa eintausend Texte alle miteinander verglichen werden, um die Ähnlichkeit festzustellen, ergeben sich daraus bereits 1000 × 1000 = 1 Million Vergleiche – wenn dabei die Wörter einzeln verglichen werden, wird es noch umfangreicher. So ein Textvergleich kann etwa sinnvoll sein, um Duplikate oder automatisiert durch Bots verschickte Spammitteilungen zu entdecken. Eine solche Textvergleichsfunktion benötigt also umso mehr Zeit, je umfangreicher das Korpus ist.

Wenn die Ausführung einzelner Befehle sehr lange braucht, sollte man sich Gedanken über die Optimierung der Skripte machen. Ein erster Ansatzpunkt kann die Aufbereitung der Datengrundlage sein – wenn man weiß, dass im Korpus Kochrezepte und Nachrichtenartikel enthalten sind, müssen nur jeweils die Kochrezepte und die Nachrichtenartikel untereinander verglichen werden. Man könnte also zunächst eine Vorprüfung durchführen und die Texte aufgrund festgelegter Stichwörter in eine der beiden Gruppen einteilen. Doch auch die verwendeten Algorithmen können optimiert werden, etwa indem effizientere Datentypen wie Zahlen statt Texten oder optimierte Sortieralgorithmen eingesetzt werden.

Beim Textvergleich stößt man mitunter noch an eine weitere Grenze: Sehr lange Texte brauchen viel Arbeitsspeicher, in den die Daten für die Verarbeitung hineingeladen werden, und auch dieser ist limitiert. Der Umfang der Ressourcen, das heißt Zeit und Raum, die ein Algorithmus benötigt, wird auch als Komplexität bezeichnet (einführend König et al. 2016, S. 313 ff.). Eine lineare Komplexität bedeutet, dass der Ressourceneinsatz mit jedem weiteren Fall gleichbleibend ansteigt, weil eine einzelne Aufgabe immer die gleiche Zeit benötigt. Bei einem naiven (d. h. nicht optimierten) Textvergleich liegt dagegen bei unterschiedlich großen Korpora quadratische Komplexität vor, jeder weitere Text führt dazu, dass alle Texte mit dem neuen Text abgeglichen werden.

An Ressourcengrenzen kann man somit in Bezug auf den Input (Datensatz) oder den Throughput (angewendete Funktion) einer Analyse stoßen. Aber auch

wenn es um den Output (Visualisierung und Bericht) wissenschaftlicher Forschung bzw. die Veröffentlichung geht, spielen die Ressourcen von Computersystemen eine Rolle. Will man etwa eine interaktive Grafik *on the fly* angepasst an Einstellungen der Nutzenden auf einer Webseite generieren, so greifen mitunter viele Personen gleichzeitig auf diese Seite zu. Es wäre bei aufwendigen Darstellungen kaum hinnehmbar, wenn jede Person so lange auf die eigene Grafik warten müsste, bis alle anderen Nutzenden ihre Grafiken erhalten haben.

Wenn Sie mit einem Problem an solche Grenzen stoßen, die von einzelnen Menschen oder Computern nicht in angemessener Zeit bewältigt werden können, dann beginnt die ‚eigentliche' Welt der Computational Methods. Neben der Optimierung von Daten und Funktionen kommt in diesen Fällen vor allem ein Verfahren zur Anwendung: Parallelisierung. Im Bereich der Datenanalyse spricht man auch von Big Data, der Begriff bezeichnet im engeren Sinn Daten, die nicht in ein einzelnes Gerät passen (Cox und Ellsworth 1997), und entsprechend in verteilten und im besten Fall parallelisierten Systemen verarbeitet werden.

Praktisch kann Parallelisierung bereits auf dem eigenen Computer beginnen, aber auch über mehrere Maschinen verteilt werden, es lassen sich drei Kernkonzepte unterscheiden:

1. **Cores:** Die meisten Computer verfügen über mehrere Recheneinheiten, sogenannte Cores, die parallel arbeiten können. Ein Skript kann auf dem eigenen Computer auf mehrere Cores aufgeteilt werden. Unter Windows können Sie beispielsweise im Taskmanager (Strg + Alt + Entf) im Bereich CPU sehen, über wie viele Cores Ihr Gerät verfügt (Abb. 6.9).
2. **Cluster:** Skripte lassen sich parallel auf mehreren Computern ausführen, die über ein Netzwerk zu einem Cluster verbunden sind. Bei dieser als High Performance Computing (HPC) bekannten Technologie wird ein Skript über einen Kontrollknoten auf die (virtuellen oder physichen) Knoten (engl. *nodes*) verteilt. Eine in wissenschaftlichen Einrichtungen häufig anzutreffende Software zur Verteilung der Skripte ist Slurm.
3. **Cloud:** Bei Cloud-Computing-Anbietern wie Amazon Webservices (AWS) können flexibel Ressourcen gemietet werden, zum Beispiel einzelne Server oder zu einem Cluster verbundene virtuelle Maschinen. Die Ausrüstung der Server kann mit wenigen Mausklicks nach Bedarf umkonfiguriert werden. Solche Lösungen eignen sich nicht nur für High Performance Computing, sondern auch zum Hosten von Anwendungen und Webseiten, etwa für die Publikation interaktiver Auswertungen.

Wenn man das erste Mal in die Welt der Parallelisierung eintaucht, ist das vermutlich eine echte Herausforderung. Wir begeben uns mit den folgenden Beispielen wie Alice im Wunderland ins Rabbit Hole und tauchen Ebene für Ebene immer weiter ab. Wenn Sie beim Lesen am Ende wieder in der Wirklichkeit ankommen, sind Sie auf einem guten Weg.

6.4.1 Parallelisierung auf mehreren Cores

Die einfachste Form der Parallelisierung können Sie auf dem eigenen Computer ausprobieren. Als Ausgangspunkt dient im Beispiel eine Document-Feature-Matrix, in der die Berichterstattung von Nachrichtenseiten erfasst ist, die laut Reuters Digital News Report im Jahr 2020 am meisten genutzt wurden (Puschmann und Haim 2021).[32] Diese Document-Feature-Matrix enthält in den Zeilen die Nachrichtenmeldungen (Dokumente) und in den Spalten die Wörter der Texte (Features). In den Zellen ist jeweils eingetragen, wie häufig das Wort im Text vorkommt (Abb. 6.7; siehe Kap. 9). Dadurch wird jede Nachrichtenmeldung durch einen sogenannten Vektor, also die Zahlen in der entsprechenden Zeile, abgebildet. Auf Grundlage der Matrix kann die Ähnlichkeit der Texte berechnet werden, etwa um Duplikate zu identifizieren. Dazu müssen alle Zeilen mit allen Zeilen verglichen werden. Eine einfache Variante zur Messung der Ähnlichkeit von zwei Vektoren stellt die Kosinusähnlichkeit dar (siehe Abschn. 4.2.4). Diese Maßzahl reicht von 1, wenn die Muster in den beiden Vektoren identisch sind, bis 0, wenn sich die Vektoren vollständig unterscheiden.[33]

Die Arbeit mit diesem Datensatz benötigt nicht nur Rechenkapazität, sondern zudem eine gute Ausstattung des Arbeitsspeichers, je nach Vorgehensweise werden über 100 GB benötigt. Grundsätzlich ist es deswegen empfehlenswert, die ersten Gehversuche und die Entwicklung von Skripten an kleineren Ausschnitten von Datensätzen vorzunehmen, auch um die Wartezeit zu verkürzen. Die entwickelten

---

[32] Diese Daten wurden von Cornelius Puschman und Mario Haim zusammengestellt und über das Center for Open Science veröffentlicht, siehe Puschmann und Haim (2021; https://osf.io/uzca3/). Zusätzlich stehen Befragungsdaten und Daten zur Interaktion mit diesen Nachrichten auf Facebook zur Verfügung. Ganz herzlichen Dank an die beiden Kollegen für diese tolle Arbeit!

[33] Die Kosinusähnlichkeit reicht eigentlich von -1 bis +1. Da die Document-Feature-Matrix bei Textvergleichen aber keine negativen Werte enthält, ist die Kosinusähnlichkeit in diesem Fall immer positiv.

doc_id	russian	climate	resident	facebook	australian	researchers	deadly	hong	santa	arguing	…
1572618596	1	0	0	0	0	2	0	0	0	0	
1680872371	0	0	0	0	0	0	0	0	0	0	
1619009312	0	0	0	0	0	0	0	0	0	0	
1692788134	0	2	0	0	0	18	1	0	0	0	
1526922325	0	0	0	0	0	0	0	0	0	0	
1407825590	1	0	0	0	0	1	0	0	0	0	
1692564005	0	0	0	0	0	2	0	0	0	0	
1458226354	0	0	0	0	0	0	0	0	0	0	
1550173514	0	2	0	0	0	14	1	0	0	0	
…											

Abb. 6.7 Eine Document-Feature-Matrix. (Quelle: eigene Darstellung, Auszug aus Puschmann und Haim (2021; https://osf.io/uzca3/))

prototypischen Skripte können anschließend über Nacht oder auf einem High Performance Cluster die vollständigen Datensätze abarbeiten. Ein Auszug des Datensatzes (☛*Repositorium*) kann in R wie folgt eingelesen werden:

```
# Packages laden
library(tidyverse)
library(quanteda)
library(quanteda.textstats)

# Document-Feature-Matrix einlesen
dfm <- read_rds("usenews_small.rds")
dfm
```

Die Dokument-Feature-Matrix umfasst hier 10.000 Dokumente und 1428 Wörter. Die folgenden Skripte laufen damit sehr schnell durch, eine Parallelisierung lohnt sich dafür noch nicht. Sie können die Skripte aber anschließend auf den Gesamtdatensatz anwenden.

Im R-Package *quanteda* (Benoit et al. 2018) findet sich die Funktion `textstat_simil()`, mit der die Kosinusähnlichkeit zwischen allen Dokumenten berechnet werden kann:

```
textstat_simil(
  dfm,
  margin = "documents",
  method = "cosine",
  min_simil = 0.99
) %>%
as_tibble()
```

Der erste Parameter enthält die Matrix. Anschließend wird angegeben, dass die Dokumente (`margin`) mit der Kosinusmethode (`cosine`) verglichen und nur Paare mit einem Ähnlichkeitswert über 0,99 (`min_simil`) ausgegeben werden sollen. Mit diesem sehr hohen Schwellwert werden fast identische Dokumente ermittelt. Das Ergebnis wird im Beispiel für die bessere Lesbarkeit mit der Funktion `as_tibble()` in eine Tabelle umgewandelt, die 14 sehr ähnliche Paare ausgibt (Tab. 6.1).

Tatsächlich ist das Ermitteln der Kosinusähnlichkeit bereits ein parallelisierter Aufruf gewesen – ohne dass es direkt sichtbar war. Denn das Package *quanteda* sorgt bei einigen Funktionen selbst dafür, die auf dem Computer verfügbaren Cores bestmöglich auszunutzen. Unter Windows können Sie das beispielsweise im Task-Manager beobachten (Abb. 6.8), die Auslastung aller Prozessoren steigt beim Aufruf von `textstat_simil()` gleichzeitig an. Allerdings ist dies eine besonders komfortable Situation. Nicht alle Funktionen sind bereits für die Parallelisierung vorbereitet – insbesondere selbstgeschriebene Funktionen mit zusätzlichen Aufbereitungsschritten – und auch, wenn die Berechnung auf ein Computercluster übertragen werden soll, sind weitere Schritte nötig.

Tab. 6.1 Kosinusähnlichkeit von Nachrichten

doc1_id	doc2_id	cosine	doc1_title	doc2_title
1564146632	1591774785	1,00	China coronavirus: Misinformation…	China coronavirus: Misinformation …
1692788134	1550173514	0,99	13 million-year-old bite marks on …	Mysterious Ice Age structure made from…
1668919094	1550173514	0,99	Humans may have arrived in North …	Mysterious Ice Age structure made from…
1557814842	1560278337	0,99	Here's why the coronavirus …	Why men are more likely to contract …
1412338756	1507331645	0,99	Why Hong Kong’s Still Protesting …	Why Hong Kong’s Still Protesting …
1581033219	1683520913	0,99	Eurojackpot - 90 Millionen…	Eurojackpot - Aktuelle Gewinnzahlen…
…				

Quelle: eigene Aufbereitung auf Grundlage von Puschmann und Haim (2021; https://osf.io/uzca3/)

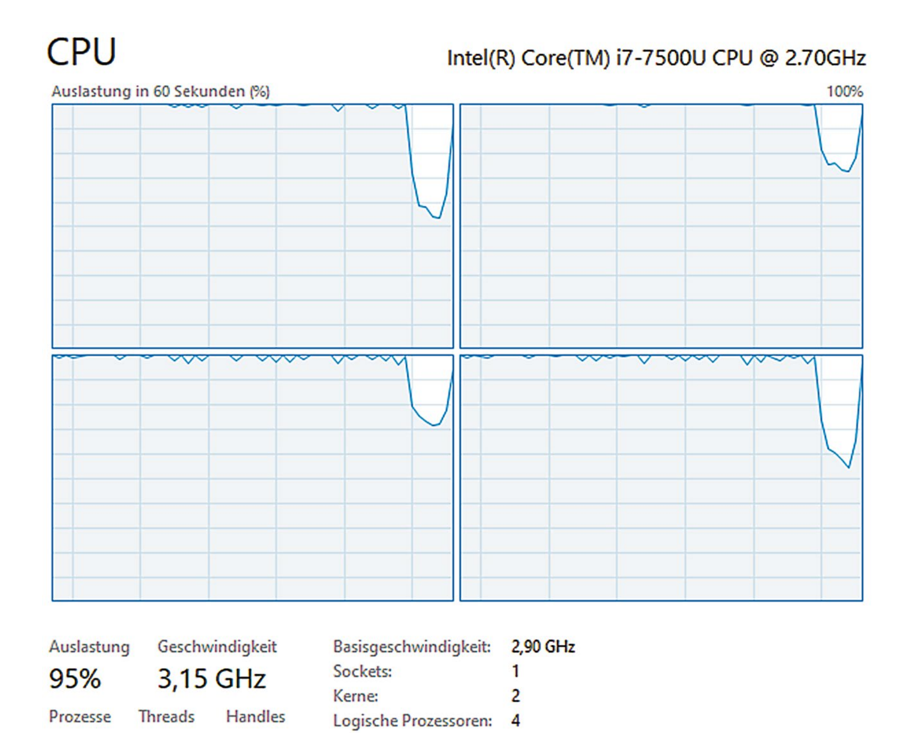


Abb. 6.8 Die Auslastung der Kerne im Windows Taskmanager. (Quelle: eigene Darstellung)

Um einen Vorgang zu parallelisieren, muss die Gesamtaufgabe in Teile zerlegt werden. Eine Vorgehensweise kann dabei sein, immer nur wenige Zeilen mit der gesamten Matrix zu vergleichen und die Teilergebnisse am Ende wieder zusammenzuführen. Dazu muss die Matrix zunächst unterteilt werden. Wenn insgesamt 10.000 Dokumente vorliegen, können daraus beispielsweise 10 Chunks mit jeweils 1000 Dokumenten gebildet werden. Der folgende Befehl erzeugt zunächst einen Vektor, in dem für jedes der Dokumente zufällig festgelegt ist, zu welchem Chunk es gehören soll. Dazu werden 10.000 Zahlen zufällig aus dem Bereich 1-10 gezogen:[34]

[34] Anstelle der Angabe 10.000 kann mithilfe von `nrow(dfm)` auch die Länge der Matrix und somit die Anzahl der Dokumente ermittelt werden. Diese Vorgehensweise ist flexibler in Bezug auf Änderungen in der Document-Feature-Matrix. Alternativ kann die Einteilung auch systematisch erfolgen, sodass die ersten 1000 Dokumente in Chunk 1, die nächsten 1000 in Chunk 2 usw. landen.

```
chunks <- sample(10, 10000, replace = TRUE)
```

Diese Zuteilung kann nun dazu genutzt werden, für jedes Chunk eine eigene Liste zu erzeugen.:

```
chunks <- split(c(1:10000), chunks)
```

Die resultierende Liste besteht entsprechend der Chunkanzahl aus zehn Elementen. Jedes der Elemente ist ein Vektor mit den Nummern der zugehörigen Dokumente.

Nach der Aufteilung der Fälle muss nun der Aufruf der Ähnlichkeitsberechnung so umgestaltet werden, dass die Funktion für jedes Chunk extra abläuft. Um eine Funktion auf eine Liste anzuwenden, gibt es mehrere Varianten (siehe Abschn. 5.1) – die Funktion `map_dfr()` bietet den Vorteil, dass die Teilergebnisse automatisch wieder zu einem Gesamtdatensatz zusammengebunden werden. Übergibt man zunächst die Liste der Chunks, wird die im zweiten Parameter übergebene Funktion für jedes einzelne Chunk ausgeführt. Die im Beispiel eingebettete Funktion[35] `function(chunk)` erhält als ersten Parameter den Chunk, für den sie jeweils zuständig ist. Der Aufruf von `textstat_simil()` unterscheidet sich von der oben angeführten Variante dahingehend, dass die Matrix zweimal übergeben wird, einmal vollständig und einmal nur mit den im Chunk definierten Zeilen. So wird der Vergleich beim Aufruf der Funktion nur für die ausgewählten Zeilen durchgeführt. Indem aber alle Chunks nacheinander abgearbeitet werden, sind am Ende alle Zeilen berücksichtigt worden:

```
map_dfr(
  chunks,
  function (chunk) {
    textstat_simil(
      dfm, dfm[chunk,],
      margin = "documents",
      method = "cosine",
      min_simil = 0.9
    ) %>% as_tibble()
  }
)
```

[35] Im Beispiel wird direkt an der benötigten Stelle eine anonyme Funktion definiert, um den Aufruf von `textstat_simil()` zu kapseln. Die Kapselung ermöglicht es, nicht nur `textstat_simil()`, sondern anschließend auch `as_tibble()` aufzurufen. In R-Funktionen wird immer das Ergebnis des letzten Aufrufs zurückgegeben.

Bislang hat noch keine explizite Parallelisierung stattgefunden, die Abwandlung des Funktionaufrufs diente nur der Vorbereitung. Der Schritt in die Erzeugung von Parallelwelten ist nun aber nahezu trivial. Die Funktion `map_dfr()` kann dazu einfach gegen die Funktion `future_map_dfr()` aus dem *furrr*-Package (Vaughan und Dancho 2022) ausgetauscht werden. Mit der `plan()`-Funktion wird festgelegt, auf wie viele Worker die Aufgabe verteilt werden soll. Unter einem Worker versteht man einen eigenständigen Prozess, der in diesem Fall auf einem eigenen Prozessorkern läuft und nacheinander die ihm automatisch zugeteilten Chunks abarbeitet.

```
library(furrr)
plan(multicore, workers = 4)

future_map_dfr(
  chunks,
  function (chunk) {

    textstat_simil(
      dfm, dfm[chunk, ],
      margin = "documents",
      method = "cosine",
      min_simil = 0.9
    ) %>%
    as_tibble()

  }
)
```

Für R und auch für Python gibt es vielfältige Packages und Varianten zur Parallelisierung. Da sich die Möglichkeiten immer weiterentwickeln, lohnt sich eine eigene Recherche. Das Grundprinzip aber bleibt immer gleich: Eine Aufgabe wird in Teile zerlegt und an Worker verteilt, nach dem Abarbeiten werden die Ergebnisse wieder in einem gemeinsamen Datensatz abgelegt.

6.4.2 Parallelisierung auf einem Cluster

Für besonders umfangreiche Berechnungen können die Worker auf mehrere Computer verteilt werden, auf denen jeweils wieder mehrere Kerne zur Verfügung stehen. Das High-Performance-Computing-Cluster (HPC) der Universität Münster umfasst beispielsweise aktuell über 15.000 Kerne verteilt auf mehrere Hundert

Knoten, so werden die einzelnen Computer genannt (Universität Münster 2022). Sofern eine Datenanalyse parallelisiert werden kann, lassen sich damit Berechnungen innerhalb eines Tages durchführen, für die sonst Jahre vergehen würden. Damit dies möglich ist, müssen die Einrichtung des Clusters, Skripte mit den Befehlen zur Berechnung und Anweisungen zum Verteilen der Aufgaben auf dem HPC zusammenspielen. Einen exemplarischen Durchlauf durch diesen Prozess finden Sie im ☛ *Repositorium* des Buchs. Wichtige Kernpunkte und Konzepte werden nachfolgend eingeführt.

Bei der Verteilung auf ein Cluster ist insbesondere zu bedenken, wie die Daten zugeteilt und wieder eingesammelt werden. Prinzipiell könnten die verschiedenen Worker miteinander kommunizieren, um sich dabei zu koordinieren. Das ist bei der Berechnung der Textähnlichkeit allerdings nicht nötig, da jeder Worker unabhängig von den anderen einen Teildatensatz bearbeiten kann. Stattdessen reicht es in solchen Fällen, wenn alle Worker auf einen gemeinsamen Ordner mit dem Gesamtdatensatz zugreifen können. Der Ablauf kann wie folgt gestaltet werden:

1. Der gesamte Datensatz wird in einen Ordner auf dem Cluster übertragen.
2. Das Skript wird auf das Cluster übertragen und auf jedem Knoten mit einer eigenen Aufgabennummer gestartet. Aus der Aufgabennummer wird abgeleitet, welchen Teildatensatz dieser Knoten bearbeiten soll.
3. Das Skript bearbeitet auf jedem Knoten seine Teilaufgabe und speichert das Ergebnis im gemeinsamen Ordner ab. Die Ergebnisse dürfen sich nicht überschreiben, deshalb wird die Nummer der Aufgabe in den Dateinamen aufgenommen.
4. Sobald alle Teilergebnisse vorliegen, können sie auf den lokalen Computer übertragen und mit einem weiteren Skript zusammengeführt werden. Alternativ kann man sich auf einen einzelnen Knoten des Clusters einloggen und die Zusammenführung dort vornehmen, um dann das Gesamtergebnis auf den eigenen Computer zu holen.

Da sich die Rahmenbedingungen einzelner Cluster durchaus unterscheiden, lässt sich kein allgemeingültiges Beispielskript konstruieren. Die Clusterbetreibenden sind aber in der Regel hilfsbereit, veröffentlichen einführende Materialien oder bieten Schulungen an. Das folgende Schema (☛ *Repositorium*) dürfte auf einer Vielzahl von Clustern funktionieren, sofern diese passend eingerichtet sind und zur Verteilung der Aufgaben Slurm[36] einsetzen. Slurm ist eine weitverbreitete Open-Source-Software zum Management von High Performance Clustern, die auch bei kommerziellen Cloudanbietern wie Amazon Web Services verwendet werden kann.

[36] Siehe SchedMD (2021, https://slurm.schedmd.com/).

Um auf einem solchen Cluster ein Skript zu starten, öffnet man auf dem eigenen Computer eine Kommandozeile und loggt sich über das Internet per SSH auf dem Cluster ein:[37]

```
ssh username@palma.uni-muenster.de
```

Die anschließend eingegebenen Befehle werden auf dem Kontrollknoten des Clusters ausgeführt. Achtung: Der Kontrollknoten ist in der Regel nicht für Berechnungen vorgesehen, wenn er blockiert wird, können die anderen Nutzenden das HPC nicht mehr erreichen. Deshalb wird auf dem Kontrollknoten lediglich ein Slurm-Befehl abgesetzt, der ein vorbereitetes Shellskript auf die anderen Knoten verteilt, welches die dort auszuführenden Befehle enthält:

```
sbatch ~/slurm_job.sh
```

In diesem Shellskript, im Beispiel *slurm_job.sh*, sind zum einen die Einstellungen für Slurm enthalten und zum anderen die Befehle, die vom Worker ausgeführt werden sollen:

```
#!/bin/bash

#SBATCH --array=0-9          # Anzahl der Tasks
#SBATCH --cpus-per-task=1    # Anzahl der Cores je Task
#SBATCH --mem-per-cpu 32G    # Benötigter RAM je Task
#SBATCH --partition=normal   # Teilbereich des Clusters
#SBATCH --time 00:60:00      # Maximale Laufzeit
#SBATCH --oversubscribe      # Knoten werden geteilt

Rscript ~/slurm_rscript.R
```

Dieses etwas gekürzte Skript erstellt einen sogenannten Array-Job, bei dem insgesamt 10 Aufgaben auf dem Cluster verteilt werden. Slurm sorgt entsprechend der

[37] Das Loginverfahren unterscheidet sich zwischen verschiedenen Clustern insbesondere dahingehend, wie das Passwort abgefragt wird. Häufig werden dazu keine Passwörter, sondern kryptografische Schlüssel eingesetzt. Ein solcher Schlüssel kann auf der Kommandozeile mit dem Befehl `ssh-keygen` erzeugt werden und besteht aus einem öffentlichen und einem privaten Schlüssel. Der öffentliche Schlüssel wird auf dem Server hinterlegt und beim Einloggen authentifiziert man sich mit dem privaten Schlüssel. Dieses Verfahren ist auch zur Authentifizierung bei anderen Diensten wie beispielsweise GitHub verbreitet.

Angaben dafür, dass für jeden Worker ein Kern und 32GB Arbeitsspeicher zur Verfügung stehen. Mit einem solchen Array-Job und der Oversubscribe-Einstellung können auch Kerne genutzt werden, die andere Nutzenden des Clusters gerade nicht benötigen. Die letzte Zeile sorgt dafür, dass auf dem Knoten R gestartet und das Skript *slurm_rscript.R* ausgeführt wird.

Innerhalb des R-Skripts kann man sich zunutze machen, dass jeder Task des Array-Jobs eine eigene Nummer erhält, die im Beispiel von 0 bis 9 reicht. Diese Task ID kann aus den sogenannten Umgebungsvariablen ausgelesen werden, um daraus das zugeordnete Chunk zu bestimmen. Mit der folgenden Berechnung bearbeitet der erste Worker die Zeilen 1 bis 1000, der zweite die Zeilen 1001 bis 2000 und so weiter, obwohl alle das gleiche Skript ausführen:

```
task_id <- as.numeric(
    Sys.getenv('SLURM_ARRAY_TASK_ID')
)

chunk_start <- (task_id * 1000) + 1
chunk_end <- chunk_start + 1000 - 1
```

Anschließend wird der Datensatz geladen, bearbeitet und das Ergebnis wird im Ordner *output* abgespeichert:

```
dfm <- read_rds("usenews_small.rds")
sim <- textstat_simil(
    dfm,
    dfm[c(chunk_start:chunk_end), ],
    margin = "documents",
    method = "cosine",
    min_simil = 0.99
)

sim %>%
  as_tibble()%>%
  write_rds(paste0("output/sim_", task_id))
```

Haben die Worker ihre Arbeit beendet, dann loggt man sich auf einem einzelnen Knoten (nicht dem Kontrollknoten!) ein oder überträgt die Teilergebnisse auf den eigenen Computer. Dort können die Liste der relevanten Dateinamen mit `dir()` erstellt, die Dateien mit `readRDS()` eingelesen und mit `map_dfr()` zusammengefügt sowie schließlich mit `write_rds()` abgespeichert werden:

```
files <- dir(
  "output",
  pattern = "sim_.*\\.rds",
  full.names = TRUE
)

sim <- map_dfr(files, readRDS)
write_rds(sim, "sim.rds")
```

Bei der praktischen Umsetzung gibt es viele Stellschrauben, um die Beispiele zu verbessern. Vor der ersten Benutzung müssen etwa die benötigten Packages auf dem Cluster installiert werden und um dies zu vereinfachen, können Container[38] eingesetzt werden (siehe Abschn. 6.3 und ☛ *Repositorium*).

6.4.3 Hosting in der Cloud

Während das High Performance Computing auf umfangreiche Berechnungen ausgerichtet ist und Wartezeiten verkürzt, gibt es neben erweiterten Ressourcen viele weitere Gründe zur Nutzung von Cloud Computing. Dafür finden sich je nach Spezialisierung unterschiedliche Typen von Cloud-Anbietern.

Kommerzielle Cloud-Plattformen bieten nützliche APIs an, die sich für spezifische Aufgaben der Datenerfassung, -aufbereitung und -veröffentlichung einsetzen lassen. Onlinedienste wie Amazons Web Services[39] können etwa dazu verwendet werden, Tracking-Daten zu sammeln, indem beim Aufruf einer URL ein Log-Eintrag in einer Datenbank erzeugt wird. Auch für Speech-To-Text, die automatisierte Bilderkennung oder zur Bereitstellung großer Datenmengen kann auf spezialisierte Funktionalitäten zurückgegriffen werden.

Sollen zudem Forschungsergebnisse in einem Dashboard veröffentlicht werden, wird ein entsprechend spezialisierter Hosting-Anbieter benötigt. Dashboards sind interaktive Anwendungen, die zur Visualisierung von Daten eingesetzt werden. Sie erlauben den Nutzenden sowohl einen schnellen Einblick in die Daten als auch die eigene Exploration. Auf diese Weise können beispielsweise Ereignisse wie die Berichterstattung auf einem Zeitstrahl oder einer Karte dargestellt werden

[38] Eine auf HPC-Clustern anzutreffende Container-Software ist Singularity. Hiermit lassen sich beispielsweise vollständig eingerichtete R-Container auf einem Cluster nutzen (Boettinger et al. 2022; https://www.rocker-project.org/).

[39] Zum Beispiel über AWS Lambda (Amazon Web Services 2022b; https://aws.amazon.com/de/lambda/).

und auch Netzwerke lassen sich visualisieren. Zu bedenken ist dabei, dass in der Regel mehrere Nutzende gleichzeitig auf eine Webseite zugreifen sollen. Anders als beim Ausleihen von Büchern aus einer Bibliothek, warten sie nicht aufeinander. Die Optimierung von Wartezeiten betrifft damit nicht nur die Durchführung einer Analyse, sondern auch den Zugriff auf die Ergebnisse einer Analyse. Als Daumenregel gilt seit langer Zeit, dass Nutzende bei der Interaktion mit einem Computersystem eine Verzögerung von bis zu einer Zehntelsekunde noch als Echtzeit empfinden (Nielsen 1993, S. 135). Um dieses Ziel bei der Umsetzung von Dashboards und Webseiten auch für umfangreiche Datensätze zu erreichen, können vorberechnete Ergebnisse eingesetzt werden – anstatt den Datensatz mit allen Nachrichtenmeldungen neu auszuzählen, wird etwa die auf Tage aggregierte Zeitreihe in ein Dashboard eingebunden. Damit die Reaktionszeiten akzeptabel sind, ist darüber hinaus die ggf. parallelisierte Rechenleistung des Geräts ausschlaggebend, auf dem das Dashboard erzeugt wird.

Dashboards können direkt aus R oder Python erzeugt werden. In R wird dazu etwa das Paket *ShinyApp* und in Python die Software *Dash* eingesetzt. Das Grundprinzip ist jeweils gleich: In einem Skript werden Elemente der Benutzeroberfläche wie Buttons oder Balkendiagrame erzeugt. Einige der Elemente sind an Ereignisse wie das Klicken oder die Eingabe von Text gebunden, das heißt, bei Änderungen durch die Nutzenden wird eine Funktion aufgerufen, die passende Daten (zum Beispiel aus einer CSV-Datei oder einer Datenbank) ausliest und die grafischen Elemente entsprechend anpasst. Zur Verwendung von ShinyApps und Dash finden sich im Web viele gute Tutorials (☛*Repositorium*; Abb. 6.9).

Dashboards agieren in der Regel als Webserver und erzeugen eine Webseite, die im Browser auf dem eigenen Computer angezeigt wird. Ein Teil der Berechnungen findet auf dem Webserver (z. B. die Datenaufbereitung über R) und ein Teil direkt im Browser der Nutzenden statt (z. B. die Visualisierung mit JavaScript), wodurch die Last verteilt wird. Während der Entwicklung wird der eigene Computer als lokaler Server eingesetzt. Um ein solches Dashboard zu veröffentlichen, kann man prinzipiell den eigenen Computer für Zugriffe aus dem Internet öffnen. Dies ist aber schon aus Sicherheitsgründen kaum zu empfehlen. Insbesondere wenn hohe Zugriffszahlen zu erwarten sind, greift man besser auf spezialisierte Hoster zurück, die ggf. auch viele parallele Zugriffe managen. Für Dashboards eignen sich etwa Anbieter wie Shinyapps.io oder Plotly.[40]

Über spezialisierte Dienste hinaus stellen Full-Stack-Anbieter die Infrastruktur für alle Arten von Serveranwendungen zur Verfügung. Hier können virtuelle Maschinen konfiguriert und (Docker-)Container bereitgestellt werden. Diese Dienste

[40] Siehe RStudio (2020; https://www.shinyapps.io/) und Plotly (2022; https://plotly.com/).

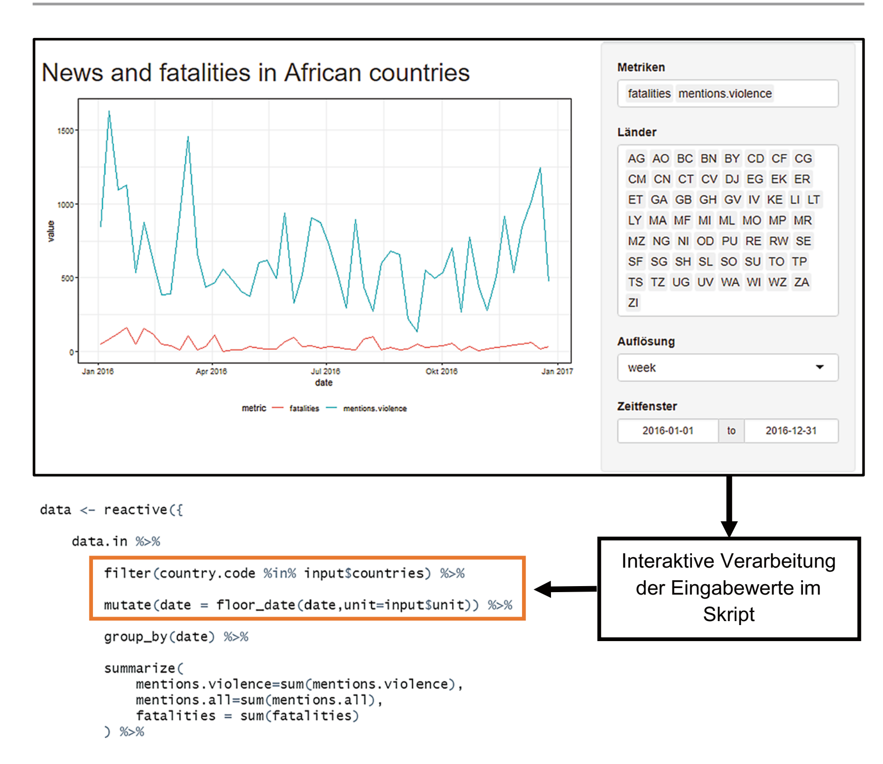


Abb. 6.9 Auszug aus einer ShinyApp. Anmerkung: Eine Veränderung der Eingabewerte führt zur Aktualisierung der Grafik. (Quelle: eigene Darstellung, ☛ *Repositorium*)

haben in der Regel für kleinere Projekte kostenlose Einstiegsangebote und umfangreiche Dokumentationen im Programm und lassen sich bei Bedarf hochskalieren. Solche Cloud-Plattformen werden insbesondere von den großen Technologiekonzernen Amazon, Microsoft oder Google bereitgestellt. Daneben finden sich auch lokale kommerzielle Dienste, die auf Cloud Computing spezialisiert sind, beispielsweise der deutsche Anbieter Hetzner. Dabei sollte man die Kosten im Blick behalten oder Kostendeckel einrichten – schließlich berechnen Hoster Geld dafür, dass sie ihre Computer zur Verfügung stellen. Deshalb lohnt es sich, die unterschiedlichen Dienste und deren Funktionen, Nutzungs- und Datenschutzbedingungen sowie Preise zu vergleichen. Universitäten stellen für wissenschaftliche Projekte häufig kostenfreie Dienste bereit und sind auch aufgrund des Datenschutzes in der Regel zu bevorzugen.

Übungsfragen

1. Was versteht man unter einem Kern und was unter einem Knoten?
2. Welche Fragestellungen fallen Ihnen ein, bei denen die Parallelisierung auf einem Cluster nötig sein könnte?
3. Recherchieren Sie, ob an Ihrer Hochschule ein HPC mit Slurm zur Verfügung steht!
4. Wie kann ein interaktives Dashboard erstellt werden?

Weiterführende Literatur

Chacon, S. & Straub, B. (2022). *Pro Git. Everything you need to know about Git.* (2. Aufl.). Berkeley: Apress. https://git-scm.com/book/en/v2

Martin, R. C. (2009). *Clean Code: Refactoring, Patterns, Testen und Techniken für sauberen Code*. Hamburg: MITP.

Öggl, B. & Kofler, M. (2018). *Docker. Das Praxisbuch für Entwickler und DevOps-Teams.* Bonn: Rheinwerk Verlag.

Santacroce, F. (2015). *Git essentials. Create, merge, and distribute code with Git, the most powerful and flexible versioning system available.* Birmingham: Packt Publishing.

Santacroce, F., Olsson, A., Voss, R. & Narębski, J. (2016). *Git. Mastering version control.* Birmingham: Packt Publishing.

Trilling, D. & Jonkman, J. G. F. (2018). Scaling up Content Analysis. *Communication Methods and Measures*, 12(2-3), 158–174. https://doi.org/10.1080/19312458.2018.1447655

van Rossum, G., Warsaw, B. & Coghlan, N. (2013). *PEP 8 – Style Guide for Python Code*. Zugriff am 25.04.2022. https://peps.python.org/pep-0008/

Vasavada, N. (2021). *Cracking Containers with Docker and Kubernetes.* Delhi: BPB Publications.

Literatur

Alpine Linux Development Team. (2022). Alpine Linux (Version 3.15.4) [Computer software]. https://alpinelinux.org/

Amazon Web Services. (2022b). *AWS Lambda*. Zugriff am 03.05.2022. https://aws.amazon.com/de/lambda/

AnsibleWorks. (2022). Ansible Automation Platform (Version 2.1) [Computer software]. https://www.ansible.com/

Apache Software Foundation. (2022b). Apache HTTP Server (httpd) (Version 2.4.53) [Computer software]. https://httpd.apache.org/

Atlassian. (2022). *Become a git guru*. Zugriff am 25.04.2022. https://www.atlassian.com/git/tutorials

Bayer, M. (2012). Sqlalchemy. In G. Wilson & A. Brown (Hrsg.), *The Architecture of Open Source Applications. Volume II. Structure, scale and a few more fearless hacks.* Mountain View: aosabook.org.

Becker, A. (2022). HeidiSQL (Version 12.0) [Computer software]. www.heidisql.com/

Benoit, K., Watanabe, K., Wang, H., Nulty, P., Obeng, A., Müller, S. et al. (2018). quanteda: An R package for the quantitative analysis of textual data. *The Journal of Open Source Software*, *3*(30), 774. https://doi.org/10.21105/joss.00774

Boettinger, C., Eddelbuettel, D. & Ross, N. (2022). *The Rocker Project. Docker Containers for the R Environment*. Zugriff am 03.05.2022. https://www.rocker-project.org/

Canonical. (2022). Ubuntu [Computer software]. https://ubuntu.com/

Cox, M. & Ellsworth, D. (1997, 19. Oktober). Application-controlled demand paging for out-of-core visualization. In *Proceedings of the 8th conference on Visualization '97 (VIS '97)* (S. 235–244). Washington DC: IEEE Computer Society Press.

DBeaver. (2022). DBeaver. Free Universal Database Tool (Version 22.1.1) [Computer software]. https://dbeaver.io/

Digital Ocean. (2022). Scotch Box (Version 3.5) [Computer software]. https://github.com/scotch-io/scotch-box

Docker. (2021). What can we help you find? Zugriff am 25.04.2022. https://docs.docker.com/

Docker. (2022a). Docker Desktop (Version 4.7.1) [Computer software]. docker.com/products/docker-desktop/

Docker. (2022b). *Docker Hub*. Zugriff am 25.04.2022. https://hub.docker.com/

Docker. (2022c). *Reference documentation*. Zugriff am 05.07.2022. docs.docker.com/reference

Eclipse Foundation. (2022). Eclipse IDE. The Leading Open Platform for Professional Developers (Version 2022-06) [Computer software]. https://eclipseide.org/release/

Git. (2022a). Git. Fast-version-control (Version 2.36.0) [Computer software]. https://git-scm.com/downloads

Git. (2022b). *Documentation – Reference*. Zugriff am 25.04.2022. https://git-scm.com/docs

GitHub. (2022b). *Where the world builds software*. Zugriff am 25.04.2022. https://github.com/

GitHub. (2022c). GitHub Desktop (Version 2.9.15) [Computer software]. https://desktop.github.com/

GitLab. (2022). *The DevOps Platform has arrived.* Zugriff am 25.04.2022. https://about.gitlab.com/

HashiCorp. (2022a). Vagrant (Version 2.2.19) [Computer software]. https://www.vagrantup.com/

HashiCorp. (2022b). *Vagrant Documentation*. Zugriff am 05.07.2022. www.vagrantup.com/docs

JetBrains. (2022a). PyCharm. The Python IDE for Professional Developers (Version 2022.1.3) [Computer software]. www.jetbrains.com/pycharm

JetBrains. (2022b). *The State of Developer Ecosystem 2021*. Zugriff am 16.01.2023. https://www.jetbrains.com/lp/devecosystem-2021/

Jünger, J. & Keyling, T. (2015). *Facepager commit e570de4, bug fix for downloaded files.* https://github.com/strohne/Facepager/commit/897d000747ec0b4b5d393d0bff86a89a59d617f0#diff-40fe3da5e33280189b65195d7cb0220d

Jupyter. (2022). JupyterLab. A Next-Generation Notebook Interface [Computer software]. https://jupyter.org/

König, L., Pfeiffer-Bohnen, F. & Schmeck, H. (2016). *Theoretische Informatik – ganz praktisch*. Berlin: De Gruyter Oldenbourg. https://doi.org/10.1515/9783110412086

Kunze, M. (1998). Laßt es leuchten. LAMP: ein datenbankgestütztes Web-Publishing-System mit Freeware. *c't*, (12), 230–231.

MariaDB Foundation. (2022). MariaDB Server. The open source relational database (Version 10.9.0) [Computer software]. https://mariadb.org/

Matsubara, Y. & GitHub contributors. (2016). *Welcome to PyMySQL's documentation!* Zugriff am 11.07.2022. Verfügbar unter: https://pymysql.readthedocs.io/

Menge-Sonnentag, R. (2021, 11. März). Nichtrassistische Sprache: Python und GitLab schicken Master aufs Abstellgleis. *Heise Online*. https://www.heise.de/news/Nichtrassistische-Sprache-Python-und-GitLab-schicken-Master-aufs-Abstellgleis-5077505.html

Microsoft. (2022b). Visual Studio Code (Version 1.67.1) [Computer software]. https://code.visualstudio.com/

Microsoft. (2022c). Windows Subsystem for Linux (WSL) (Version 2) [Computer software]. https://docs.microsoft.com/de-de/windows/wsl/

Nielsen, J. (1993). *Usability engineering*. Boston: AP Professional.

Ooms, J., James, D., DebRow, S., Wickham, H., Horner, J. & RStudio. (2021). RMySQL. Database Interface and 'MySQL' Driver for R (Version 0.10.23) [Computer software]. https://cran.r-project.org/package=RMySQL

Oracle. (2022c). VirtualBox (Version 6.1.34) [Computer software]. https://www.virtualbox.org/

Plotly. (2022). *The front end for ML and data science models*. Zugriff am 03.05.2022. https://plotly.com/

Puschmann, C. & Haim, M. (2021). useNews. [Dataset]. Zugriff am 28.01.2022. https://osf.io/uzca3/

RStudio. (2020). *Shinyapps.io*. Zugriff am 03.05.2022. https://www.shinyapps.io/

RStudio. (2022a). Download the RStudio IDE. Zugriff am 25.04.2022. https://www.rstudio.com/products/rstudio/download/

Santacroce, F. (2015). *Git essentials. Create, merge, and distribute code with Git, the most powerful and flexible versioning system available*. Birmingham: Packt Publishing.

SchedMD. (2021). *Slurm workload manager. Documentation*. Zugriff am 03.05.2022. https://slurm.schedmd.com/

Spyder project contributors. (2022). Spyder. The Scientific Python Development Environment (Version 531) [Computer software]. https://www.spyder-ide.org/

The PHP Group. (2022). PHP (Version 8.1.5) [Computer software]. https://www.php.net/

Universität Münster. (2022). *High Performance Computing*. Zugriff am 03.05.2022. https://www.uni-muenster.de/IT/services/unterstuetzungsleistung/hpc/

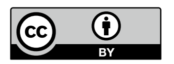
Van Dijck, J. (2020). Seeing the forest for the trees: Visualizing platformization and its governance. *New Media & Society*, *23*(9), 2801–2819. https://doi.org/10.1177/1461444820940293

Van Rossum, G., Warsaw, B. & Coghlan, N. (2013). *PEP 8 – Style Guide for Python Code*. Zugriff am 25.04.2022. https://peps.python.org/pep-0008/

Vaughan, D. & Dancho, M. (2022). *furrr: Apply Mapping Functions in Parallel using Futures*. Zugriff am 18.01.2023. https://furrr.futureverse.org/

W3Techs. (2022). *Usage statistics of PHP for websites*. Zugriff am 05.07.2022. w3techs.com/technologies/details/pl-php

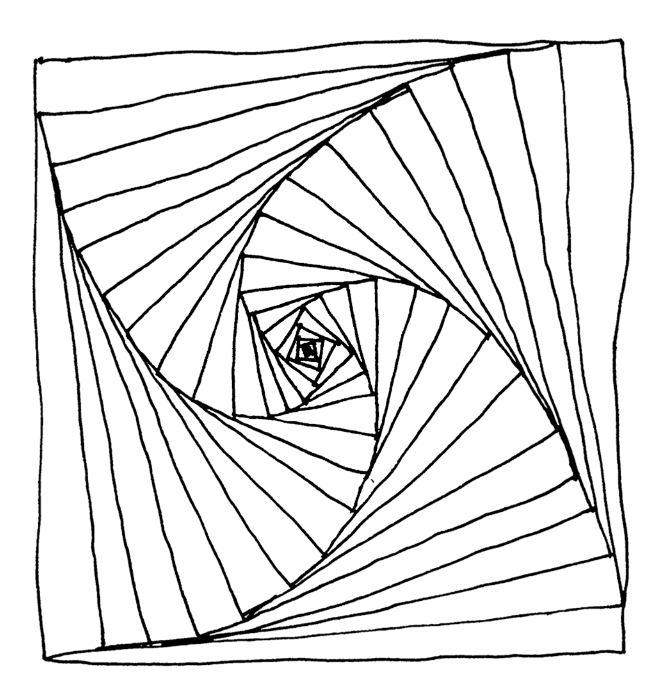
Wickham, H., Girlich, M. & Ruiz, E. (2022). dbplyr. A 'dplyr' back end for databases [Computer software]. https://dbplyr.tidyverse.org/


BY

Teil III

Anwendungsfelder

7 Automatisierte Datenerhebung



J. Jünger, C. Gärtner, *Computational Methods für die Sozial- und Geisteswissenschaften*, https://doi.org/10.1007/978-3-658-37747-2_7

Zusammenfassung

Dieses Kapitel führt in die automatisierte Datenerhebung mittels Webscraping und über Application Programming Interfaces (APIs) ein. Sie lernen, wie Sie mit Python oder R einen Webscraper entwickeln und wie Sie mit APIs interagieren können.

Im Online-Repositorium unter https://github.com/strohne/cm finden Sie begleitend zum Kapitel weitere Materialien, auf die wir im Text mit ☛ verweisen.

Schlüsselwörter

Webscraping · Webcrawling · Beautiful Soup · rvest · Browser Automation · Selenium · Application Programming Interface (API) · HTTP-Statuscode · Facepager · Social Media · Google Cloud Vision

Automatisierte Verfahren der Datenerhebung spielen vor allem dort eine Rolle, wo sich das Handeln von Akteuren in Daten niederschlägt.[1] Das betrifft insbesondere die internetvermittelte Kommunikation, etwa wenn Anbieter Webseiten erstellen und die Zugriffe protokollieren, wenn Nutzer:innen auf diesen Webseiten Inhalte veröffentlichen, kommentieren und bewerten oder wenn Forscher:innen ihre Ergebnisse auf Webseiten publizieren. Steht man als Wissenschaftler:in einem Webangebot gegenüber, so lassen sich drei Ebenen unterscheiden, auf denen sich das Ergebnis von Handeln wiederfindet (siehe auch Kap. 2). In seltenen Fällen bekommt man Zugriff auf die Ebene der Datenbanken, aus denen Webseiten erzeugt und in denen Zugriffe protokolliert werden. Dahingegen stellen jedoch einige Anbieter Programmierschnittstellen bereit, über die ein automatisierter Zugang zu Funktionen und vorstrukturierten Daten möglich ist. Auch ohne spezielle Schnittstellen können Daten direkt aus Webseiten ausgelesen werden, denn auf Ebene der Browser sind Texte und Nutzungsindikatoren für Wissenschaftler:innen genauso sichtbar wie für andere Nutzer:innen. In den folgenden Kapiteln geht es darum, wie solche Daten mit eigenen Skripten erhoben werden.

Die Ausführungen setzen erste Erfahrungen mit Python und R voraus (siehe Kap. 5). Hilfreich sind auch grundlegende Kenntnisse über HTML und URLs

[1] Zu bedenken ist bei der Interpretation, dass diese Artefakte auch durch automatische Verfahren (Bots) hervorgerufen werden können.

(siehe Kap. 2 und 3). Es geht hier einerseits darum, die grundsätzliche Vorgehensweise zu vermitteln und ein Grundgerüst zur Verfügung zu stellen. Andererseits werden Hinweise auf typische Hürden gegeben. Aus diesen Zutaten können Sie anschließend mit etwas Geduld und Kreativität eigene Lösungen entwickeln.

7.1 Webscraping

Webscraping bezeichnet das automatisierte Extrahieren von Daten aus Webseiten. In einem allgemeinen Sinn kann man darunter alle Verfahren verstehen, bei denen Daten über das HTTP-Protokoll übertragen und anschließend aufbereitet werden. Das HTTP-Protokoll ist eine wesentliche Grundlage des Web und dient vor allem dazu, dass Webseiten in Webbrowsern angezeigt werden können. Es ist in der Adresszeile von Browsern erkennbar: Das Protokoll wird in der Regel bei unverschlüsselten Verbindungen in der Form „http://“ oder bei verschlüsselten Verbindungen als „https://“ direkt vor der URL angezeigt.[2]

In einem engeren Sinn versteht man unter Webscraping dagegen nur diejenigen Verfahren, bei denen Webseiten im HTML-Format verarbeitet werden (siehe Abschn. 3.4). Auch APIs bauen teilweise auf dem Web auf, geben Daten aber in strukturierter Form zurück, etwa im JSON-Format. Eine Spezialform von Webscraping ist das Webcrawling. Hierbei werden die Links einer Webseite ausgelesen und verfolgt, das heißt, es werden immer weitere Webseiten gesammelt. Webcrawler, auch als Webspider bezeichnet, arbeiten sich auf diese Weise durch das Netz. Insbesondere Suchmaschinen verwenden dieses Verfahren, um so nach und nach alle Webseiten zu erfassen.

Für Webscraping als Methode zur automatisierten Datenerhebung gibt es typischerweise drei Möglichkeiten:

1. Beim **klassischen Webscraping** erhält ein Skript eine Liste von URLs und lädt die entsprechenden Seiten herunter. Aus dem so abgespeicherten Quelltext werden die Daten extrahiert.
   Ein Vorteil dieser Methode besteht darin, dass sie vergleichsweise einfach umzusetzen ist und auch gut skaliert, das heißt für eine hohe Anzahl von Webseiten eingesetzt werden kann. Aufwendig sind allerdings Authentifizierungs-

[2] Siehe Abschn. 2.1. zum Aufbau von URLs und zu Webseiten als Datenquellen.

verfahren über Cookies (wenn man sich bei einer Webseite anmelden muss), eingeblendete Popups oder der Umgang mit dynamischen Inhalten (wenn Webseiten Daten im Hintergrund nachladen).

2. Bei der **Automatisierung des Webbrowsers** startet ein Skript einen Browser, der dann ferngesteuert wird. Einerseits kann dann wie beim klassischen Webscraping der Quelltext der Seiten abgespeichert und weiterverarbeitet werden. Andererseits sind die Inhalte der Seiten im Browser in einem sogenannten Document Object Model (DOM) verfügbar, auf das ein Skript zugreifen kann.
   Ein Vorteil dieser Vorgehensweise ist, dass sie dem manuellen Surfen im Web sehr nahekommt. Auch Webseitenbetreiber können hier kaum unterscheiden, ob ein Skript bzw. Bot oder ein Mensch auf die Webseite zugreift. Zudem kann man sich bei einer Webseite anmelden, indem Benutzername und Passwort ferngesteuert in die entsprechenden Anmeldefelder geschrieben werden oder indem man zwischen automatisierten Eingaben und manuellem Eingreifen hin und her wechselt. Auch dynamische Inhalte, die vom Browser automatisch nachgeladen werden, sind zugänglich, wenn man den Ladevorgang abwartet. Diese Vorgehensweise ist sehr langsam, hat aber den Charme, dass man dem Computer beim Surfen zuschauen kann.
3. **Spezialprogramme** wie RapidMiner, ScrapeBox oder Facepager bieten dagegen grafische Benutzeroberflächen an, um Seiten herunterzuladen und zu verarbeiten.[3]
   Als Vorteil mag hierbei erscheinen, dass man nicht programmieren muss. Wer keine Programmiererfahrung hat, bekommt hiermit den schnellsten Einstieg. Dieser Vorteil ist aber trügerisch, denn gleichzeitig gibt man die Hoheit über den Prozess der Datenerhebung auf. Nicht nur die Funktionsweise einer Plattform wie Facebook ist dann eine Black Box, sondern zusätzlich auch die zur Datenerhebung eingesetzte Software.

Alle drei Möglichkeiten werden im Folgenden kurz vorgestellt.

7.1.1 Klassisches Webscraping

Klassisches Webscraping kann in drei Schritte unterteilt werden. Zunächst werden die benötigten Webseiten heruntergeladen, das heißt, die HTML-Dateien werden abgespeichert. Im zweiten Schritt werden aus diesen Dateien die gewünschten Da-

[3] Siehe RapidMiner (2021; https://rapidminer.com/), ScrapeBox (2022; http://www.scrapebox.com/) und Jünger und Keyling (2022; https://github.com/strohne/Facepager/).

ten extrahiert. Dazu werden die Dateien in der Regel so eingelesen, dass die Struktur der Daten in Strukturen der Programmiersprache überführt wird. Ein solcher Prozess wird parsen genannt und das dazu eingesetzte Programm ist ein Parser. Schließlich werden diese extrahierten Daten in einem Format abgespeichert, das für die weitere Verarbeitung geeignet ist, etwa in Tabellen im CSV-Format. Webscraping lässt sich mit vielen verschiedenen Programmiersprachen umsetzen.[4] Mit Python (siehe Abschn. 5.2) läuft dieser Prozess typischerweise wie folgt ab (unten finden Sie ein Beispiel mit R):

1. Mit Funktionen aus der Bibliothek *requests* werden **Webseiten heruntergeladen** und auf dem Computer abgespeichert (Reitz 2020). Zusätzlich kommen in der Regel noch Funktionen aus der Bibliothek *os* zum Einsatz, um mit Dateien und Verzeichnissen umzugehen. Hierbei ist zu beachten, dass die reinen HTML-Dateien heruntergeladen werden und deshalb zum Beispiel kein JavaScript ausgeführt wird. Einige Seiten enthalten nur ein Grundgerüst und laden die Inhalte erst mit JavaScript und sogenannten XMLHttpRequests (XHR) nach. Diese Seiten können mit klassischem Webscraping nicht ohne Weiteres ausgelesen werden.

   Häufig sind die Betreiber allerdings darauf eingestellt, zumindest die wesentlichen Inhalte auch ohne JavaScript auszuliefern. Mitunter werden auch spezielle Mobilseiten angeboten, die ohne JavaScript auskommen. Um zu erkunden, wie eine Seite ohne JavaScript aussieht, kann man dieses im Browser über die Einstellungen übergangsweise deaktivieren. Einige Anbieter stellen ihre Inhalte zusätzlich als RSS-Dateien zur Verfügung. Das hat den Vorteil, dass die Daten ähnlich wie bei der Verwendung von APIs schon vorstrukturiert sind. Diese Dateien enthalten XML und können im Prinzip genauso verarbeitet werden wie HTML-Seiten.
2. Die Dokumente werden anschließend mit Funktionen aus der Bibliothek *Beautiful Soup* **geparst** (Richardson 2022). Beautiful Soup kann sowohl HTML als auch XML verarbeiten und ist je nach Modus mehr oder weniger fehlertolerant. Das ist wichtig, weil Webseiten nicht immer standardkonform sind. Genauso wie dieses Dokument ganz sicher Rechtschreib-, Formulierungs- und Formatierungsfehler enthält, wird man auch im Web regelmäßig auf Fehler oder zumindest Sonderfälle stoßen. Nach dem Einlesen liegen die Daten in einem sogenannten Document Object Model (DOM) vor, das heißt die Teile des Dokuments

[4] Webscraping ist sogar ausschließlich über die Kommandozeile möglich. Zum Herunterladen können die Programme wget oder curl genutzt werden, zum Extrahieren von Daten mittels regulärer Ausdrücke zum Beispiel grep.

sind in einer hierarchischen Baumstruktur abgelegt, wobei die Elemente Knoten genannt werden, die Verbindungen zwischen den Knoten heißen Achsen. Für den Zugriff auf die Elemente stellt Beautiful Soup mehrere Möglichkeiten bereit:

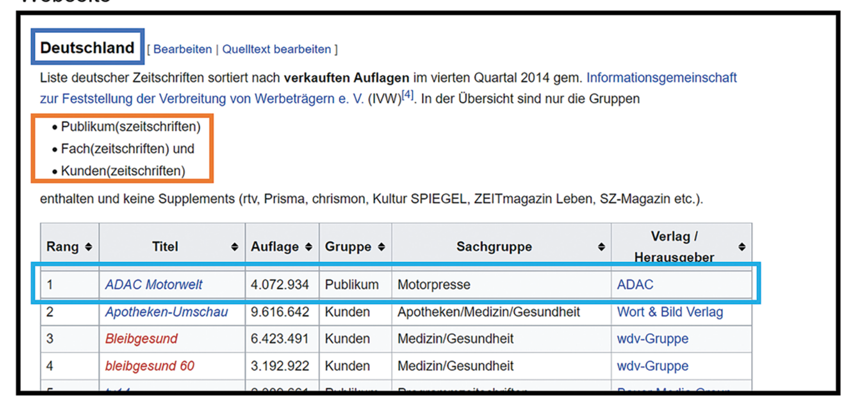
- **Navigieren**: Man kann sich im baumartig aufgebauten DOM von Ast zu Ast hangeln, wobei sich die Stammbaumterminologie eingebürgert hat. Ausgehend von einem Element kann man die Kindelemente (engl. *children*), Geschwisterelemente (engl. *siblings*) oder das Elternelement (engl. *parent*) erreichen.
- **Suchen**: Elemente können mittels ihres Namens oder ihrer Attribute gesucht werden (siehe Abschn. 3.4). Dabei kann man auch reguläre Ausdrücke verwenden (siehe Abschn. 4.1.1).
- **CSS-Selektoren**: Cascading Stylesheets (CSS) sind eigentlich eine Sprache zur Formatierung von Webseiten, zum Beispiel um Schriftfarben oder die Breite einer Spalte festzulegen. Mit Selektoren wird bestimmt, welche Elemente formatiert werden sollen. Die einfachsten Selektoren bestehen nur aus dem Namen eines Elements. Man kann Elemente zudem durch ihre Klassenattribute und IDs auswählen sowie in hierarchisch tieferliegende Ebenen des DOM eintauchen (siehe Abschn. 4.1.2).

3. Die Inhalte der so ausgewählten Elemente werden schließlich **in einer passenden Datenstruktur abgelegt**. Wenn mehrere Seiten ausgelesen werden, kann zum Beispiel eine Liste von Dictionaries verwendet werden. In jedem Dictionary sind dann die einzelnen Datenwerte einer Seite als Name-Wert-Paare abgelegt. Eine solche Datenstruktur lässt sich im JSON-Format abspeichern, sie lässt sich aber auch in eine Tabelle umformen. Jedes Element der Liste entspricht dabei einer Zeile, die Namen innerhalb der Dictionaries geben die Spalten an und die Werte in den Dictionaries entsprechen den Zellen der Tabelle. Die Bibliothek *pandas* (Pandas development team 2022a) stellt Funktionen bereit, mit denen Listen von Dictionaries in tabellenartige Strukturen umgeformt werden können. Das Ergebnis lässt sich dann zum Beispiel als CSV- oder Excel-Datei abspeichern.

Das folgende Beispiel setzt diese Schritte um und extrahiert eine Wikipedia-Tabelle deutscher Zeitschriften – sortiert nach Auflagenstärke im vierten Quartal 2014 (Abb. 7.1). Möglicherweise müssen Sie die Skripte etwas anpassen, denn das Web entwickelt sich beständig weiter und die Anbieter ändern immer wieder die Struktur von Webseiten. Die grundsätzliche Vorgehensweise bleibt aber gleich.

Erstellen Sie zunächst ein neues Jupyter-Notebook (siehe Abschn. 5.2) und gehen Sie dann Stück für Stück vor, indem Sie einzelne Zellen einfügen und ausfüh-

Webseite



Quelltext

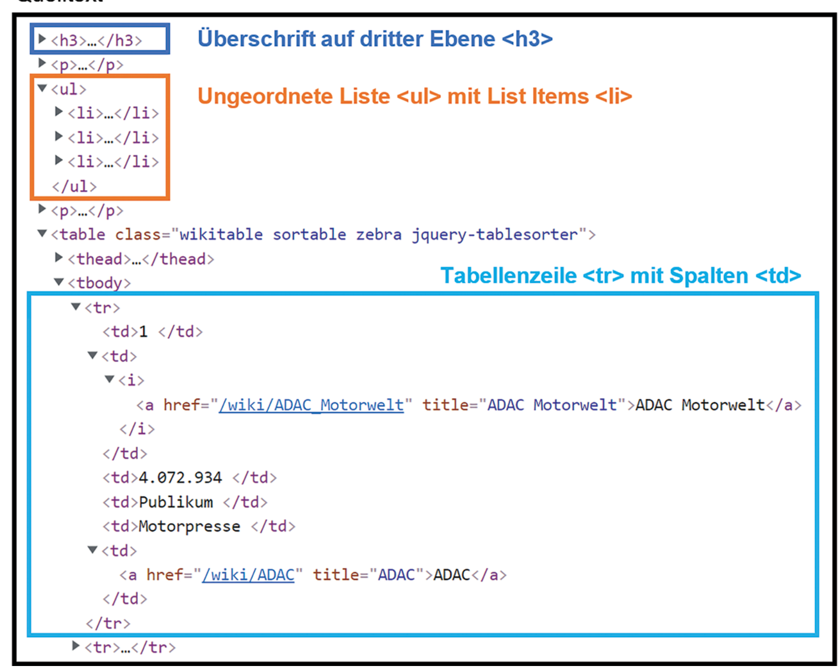


Abb. 7.1 Webseite und Quelltext einer Seite im Vergleich. (Quelle: Ausschnitt aus Wikipedia (2021, https://de.wikipedia.org/wiki/Liste_auflagenstärkster_Zeitschriften#Deutschland), CC-BY-SA. Den vollständigen Quelltext können Sie im Browser mit der Entwicklerkonsole betrachten (siehe Abschn. 3.4))

ren (☛ *Repositorium*). Versuchen Sie dabei zu verstehen, was der Code bewirkt. Zur Erinnerung: Die Zeilen mit einem Hash # am Anfang sind Kommentare, die nur der Erläuterung und Dokumentation dienen. Zur Überprüfung der Schritte können Sie den Inhalt der Objekte mit dem `print()`-Befehl ausgeben. Grundsätzlich wird in Jupyter-Notebooks das zuletzt angegebene Objekt in einer Zelle automatisch ausgegeben, sodass der `print()`-Befehl nicht immer nötig ist.

Schritt 1: Dokumente herunterladen Im ersten Schritt werden die Dokumente heruntergeladen und in einer Datei im Arbeitsverzeichnis abgespeichert. Wesentlich ist der Befehl `requests.get()`, mit dem eine Anfrage an den Webserver gesendet wird, um eine Seite herunterzuladen. Welche Seite genau heruntergeladen werden soll, wird über eine URL angegeben. Der Server antwortet auf die Anfrage zunächst mit einem Statuscode. Mit dem Statuscode 200 signalisiert der Server, dass die Seite vorhanden ist und heruntergeladen werden kann. Weitere häufig anzutreffende Statuscodes sind 404, wenn eine Seite nicht vorhanden ist, oder 500, wenn auf dem Server ein Fehler auftritt.[5]

```
# Bibliotheken einbinden
import os
import requests

# Unterverzeichnis zum Herunterladen festlegen und
# anlegen, wenn es noch nicht existiert
directory = "html"
if not os.path.exists(directory):
    os.makedirs(directory)

# URL und Dateiname festlegen
url = "https://de.wikipedia.org/wiki/" + \
      "Liste_auflagenst%C3%A4rkster_Zeitschriften"
dateiname = directory + "/zeitschriften.html"

# Herunterladen
response = requests.get(url)
```

[5] Die URL enthält einen Umlaut „ä“, der hier als Prozentkodierung angegeben ist (siehe Abschn. 2.1). Leerzeichen und Umbrüche haben in Python eine Bedeutung. Da einige Zeilen zu lang für die Darstellung sind, wurden sie über das Zeichen \ umgebrochen. Python interpretiert dieses Zeichen so, dass der Code fortgesetzt wird.

```
    if response.status_code == 200:
        with open(dateiname, 'wb') as datei:
            datei.write(response.content)
```

Nach dem Ausführen dieses Abschnitts sollte im Unterverzeichnis *html* eine HTML-Datei der Webseite liegen. Öffnen Sie die Datei mit dem Browser oder mit einem Texteditor wie Atom, um sich den Inhalt anzuschauen. In der Browseranzeige der HTML-Datei kann es Ihnen so vorkommen, als wäre die Datei kaputt, da sie nicht ordentlich formatiert ist. Dies liegt daran, dass nur der HTML-Inhalt ohne die grafische Gestaltung (CSS-Dateien) heruntergeladen wurde – für die Datenerhebung ist das in der Regel genau richtig.

Bei der Anfrage lassen sich noch weitere Informationen an den Webserver schicken, um mit einem Skript das menschliche Surfen nachzuahmen. Zum Beispiel können Cookies mitgeschickt werden, um sich bei einer Seite anzumelden. Hilfreich ist auch das Übermitteln eines User-Agent. Der User-Agent gibt an, mit welchem Programm auf die Webseite zugegriffen wird – damit kann sich ein Skript als Webbrowser wie Chrome oder Firefox ausgeben. Beim Webscraping wird das hin und wieder nötig, da einige Webserver nur Anfragen von Browsern, nicht aber Skripten, akzeptieren. Um herauszufinden, welche Cookies oder welchen User-Agent ein Browser normalerweise versendet, können Sie im Browser die Entwicklerkonsole verwenden.[6] Wie diese Angaben mitgeschickt werden können, lässt sich in der Dokumentation der *requests*-Bibliothek nachlesen.[7]

Schritt 2: Dokument parsen Im zweiten Schritt wird mit `BeautifulSoup()` die Datei eingelesen und im Objekt `soup` abgelegt. Sie können dieses Objekt beliebig benennen, es repräsentiert das Wurzelelement der Seite. Von dort kann auf das Document Object Model und damit auf die Kindelemente zugegriffen werden. Schauen Sie sich den Quelltext genau an, um die gewünschten Elemente zu identifizieren – dafür müssen Sie zunächst den Aufbau von HTML-Dateien verstehen (siehe Abschn. 3.4)! Die Namen der Elemente können Sie am einfachsten mit der Entwicklerkonsole des Browsers herausfinden. Im Beispiel soll eine Tabelle aus

[6] Die Entwicklerkonsole lässt sich etwa in Firefox durch Rechtsklick auf eine Webseite mit der Option Element Untersuchen oder durch Drücken der Taste F12 öffnen. Anschließend können Sie unter Netzwerkanalyse eine beliebige Anfrage anklicken. Es sollten die Details der Anfrage angezeigt werden wie beispielsweise die Header, unter denen auch der User-Agent gelistet ist. Ein User-Agent für Firefox lautet zum Beispiel „Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:88.0) Gecko/20100101 Firefox/88.0.“

[7] Siehe Reitz (2022; https://requests.readthedocs.io/).

dem Quellcode extrahiert werden. Tabellen sind in `<table>`-Elementen enthalten. Darin ist jede Zeile in einem `<tr>`-Tag gekapselt, worin sich wiederum die einzelnen Spalten in `<td>`-Tags finden (Abb. 7.1).

Die *Beautiful Soup*-Funktion `select()` erlaubt es, mittels CSS-Selektoren Elemente zu finden. Im Output erhält man eine Liste mit allen Elementen, auf die der CSS-Selektor zutrifft.[8] Im Beispiel werden zunächst ausgehend vom Wurzelelement mit `select('table')` alle untergeordneten Tabellen, also alle `<table>`-Elemente, aus dem DOM ausgelesen. Aus der Liste `tables` kann dann mit `[3]` auf die vierte Tabelle,[9] diejenige mit den deutschen Zeitschriften, zugegriffen werden. Innerhalb der Tabelle können nach dem gleichen Schema alle Zeilen gewonnen werden. Die erste Zeile enthält keine Inhalte, sie wird mit `table_rows[1:]` aussortiert, indem nur Zeilen ab Index 1 behalten werden. Es lohnt sich, die Dokumentation von Beautiful Soup zu lesen, um einen Überblick über weitere Möglichkeiten zu erhalten.[10]

```
# Bibliotheken einbinden
from bs4 import BeautifulSoup

# Datei öffnen und HTML parsen
soup = BeautifulSoup(
    open(dateiname, encoding="utf-8"),
    'lxml'
)

# Alle Tabellen finden, vierte Tabelle rausziehen
tables = soup.select('table')
table_de = tables[3]

# Alle Zeilen in Tabelle finden
# Die erste Zeile (ohne Inhalte) entfernen
rows = table_de.select('tr')
rows = rows[1:]
```

[8] Alternativ kann man mit `find_all()` oder `find()` auch Elemente nach Namen oder Attributen durchsuchen oder sich mit Funktionen wie `next_element()` und `previous_element()` durch den Baum navigieren.

[9] Da die Indizierung in Python bei 0 beginnt, ist die vierte Tabelle auf Position 3.

[10] Siehe Richardson (2020; https://www.crummy.com/software/BeautifulSoup/bs4/doc/).

Aus allen gefundenen Zeilen, die als Liste im neu erstellten Objekt `rows` abgelegt sind, werden anschließend die Inhalte ausgelesen. In jedem Durchlauf der For-in-Schleife werden zunächst über `select('td')` alle Spalten innerhalb einer Zeile gefunden und in der Liste `cols` abgelegt. Auf eine einzelne Spalte wird wie schon bei den Tabellen über Indizes zugegriffen. Der Text des Elements lässt sich über `.text` extrahieren und mit `.strip()` so bereinigen, dass Leerzeichen oder Markierungen von Umbrüchen am Anfang und am Ende entfernt werden.

Um die extrahierten Daten einzusammeln, wird zunächst ein leeres Dictionary `item` angelegt. Innerhalb eines Durchlaufs der For-Schleife wird das Dictionary mit Einträgen gefüllt und der Liste `results` hinzugefügt.

```
# Liste, um die Ergebnisse abzulegen
results = []

# Alle Zeilen abarbeiten...
for row in rows:

    # Alle Spalten innerhalb einer Zeile finden
    cols = row.select('td')

    # Text aus Spalten auslesen
    item = {}
    item['rang'] = cols[0].text.strip()
    item['titel'] = cols[1].text.strip()
    item['auflage'] = cols[2].text.strip()
    item['gruppe'] = cols[3].text.strip()

    # Das dict zur results-Liste hinzufügen
    results.append(item)
```

Nach dem Ausführen dieses Abschnitts liegen die extrahierten Inhalte in der Liste `results` vor. Im Beispiel werden die ersten vier Spalten der Tabelle ausgelesen. Versuchen Sie, weitere Daten auszulesen oder aufzubereiten und in der `results`-Liste abzulegen!

Hilfreich könnte es beispielsweise sein, die Links auszulesen, durch die man bei einem Klick auf den Titel zum entsprechenden Wikipedia-Artikel gelangt. Diese Links sind innerhalb der zweiten Spalte in den `<a>`-Elementen enthalten, genauer in den `href`-Attributen. Da aus jeder Spalte nur ein einzelnes Element ausgelesen werden soll, nicht eine ganze Liste, wird die `select _ one()`-Funk-

tion verwendet. So spart man sich den Zugriff über die Indizes. Aus diesem Element kann man mit `get('href')` dann die URL auslesen:

```
link = cols[1].select_one('a')
if link:
    item['link'] = "www.wikipedia.org" + \
                   link.get('href')
else:
    item['link'] = None
```

Grundsätzlich muss man sich darauf einstellen, dass innerhalb einer Website häufig nicht alle Tabellen, Listen oder Unterseiten gleich aufgebaut sind und dass sich Webseiten immer wieder ändern. Damit ein Skript nicht jedes Mal abbricht, wenn man auf solch eine Hürde stößt, baut man Fehlerbehandlungen ein. Im vorangegangenen Beispiel ist eine einfache Variante umgesetzt: Nicht alle Artikel enthalten einen Link, der Zugriff über `link.get()` würde dann zu einem Fehler führen. Deshalb wird mit einer If-else-Verzweigung geprüft, ob das Element erfolgreich ausgelesen werden konnte.

Eine universelle Möglichkeit der Fehlerbehandlung bieten Try-except-Blöcke. Innerhalb des Try-Blocks wird zunächst der Code formuliert, der ausgeführt werden soll. Wirft dieser einen Fehler aus, wird zum Except-Block gesprungen. In diesem ist festgelegt, wie alternativ verfahren wird – etwa ob die entsprechende Funktion übersprungen wird oder ein leerer Wert zurückgegeben werden soll. Bauen Sie das Beispiel einmal entsprechend um!

Schritt 3: Daten abspeichern Im letzten Schritt geht es darum, die Daten für die weitere Analyse abzuspeichern. Mit der *pandas*-Bibliothek wandeln Sie die Liste in eine Tabelle um und können diese dann als CSV-Datei abspeichern. Damit die Datei leichter mit Excel geöffnet werden kann, geben Sie das Semikolon als Trennzeichen an:

```
# Bibliotheken einbinden
import pandas as pd

# Liste mit Dictionaries in Dataframe umwandeln
results = pd.DataFrame(results)

# Vorschau des Dataframe ausgeben
display(results)
```

```
# Dataframe als CSV-Datei abspeichern
results.to_csv(
    "results.csv",
    sep=";", index=False
)
```

Wenn dieser Schritt funktioniert hat, dann sehen Sie dank der Funktion `display()` im Notebook einen Auszug aus der Tabelle mit den extrahierten Daten. Außerdem sollte im Verzeichnis von JupyterLab eine CSV-Datei liegen. Öffnen Sie die Datei mit einem Texteditor und mit einem Programm wie Excel, um den Aufbau zu verstehen.

Wenn Sie das Beispiel an andere Zwecke anpassen und weiter ausbauen, etwa um in einer Schleife mehrere Seiten nacheinander automatisch herunterzuladen, wird es schnell unübersichtlich. Sie können die Übersichtlichkeit verbessern, indem Sie Teile des Codes in eigene Funktionen verpacken (☛ *Repositorium*). Für umfangreichere Projekte lohnt sich darüber hinaus die Arbeit mit speziell dafür entwickelten Bibliotheken. Empfehlenswert ist hierfür das Framework Scrapy.[11]

Alternative: Webscraping mit R Die gleiche Vorgehensweise lässt sich auch mit R (siehe Abschn. 5.1) umsetzen. Die benötigten Bibliotheken sind *tidyverse* (Wickham 2019b) und *writexl* (Ooms und McNamara 2021) für die Verarbeitung und das Abspeichern der Ergebnisse. Über die Bibliothek *rvest* (Wickham 2022b) kann eine Seite heruntergeladen und geparst werden:[12]

```
# Packages
library(tidyverse)
library(writexl)
library(rvest)

# Seite herunterladen und parsen
url <- "https://de.wikipedia.org/wiki/↵
        Liste_auflagenst%C3%A4rkster_Zeitschriften"
html <- read_html(url)
html
```

[11] Siehe Zyte (2022; https://scrapy.org/).

[12] Aus Darstellungsgründen wurde die Zeichenkette umgebrochen und mit dem Umbruchzeichen ↵ markiert. In R sollte der Code ohne Umbruch eingegeben werden.

Das erzeugte `html`-Objekt ist vergleichbar mit dem `soup`-Objekt von Beautiful Soup. Es kann nun mit ähnlichen Funktionen auf die Elemente im Document Object Model (DOM) zugegriffen werden. Besonders einfach ist der Zugriff mit der Funktion `html_nodes()`. Diese Funktion erwartet als ersten Parameter das `html`-Objekt und als zweiten Parameter einen CSS-Selektor.[13] Um auf alle `<table>`-Elemente zuzugreifen, kann man den CSS-Selektor `table` verwenden:[14]

```
el_tables <- html_nodes(html, "table")
```

Im Sinne der Tidyverse-Logik ist besonders bei mehreren Verarbeitungsschritten der Pipe-Operator `%>%` hilfreich – etwa wenn zunächst die Elemente und im nächsten Schritt dann deren Attribute oder Inhalte extrahiert werden. Die Pipe schiebt im folgenden Beispiel das Ausgangsobjekt `html` als ersten Parameter in die Funktion `html_nodes()`. Diese Formulierung ist äquivalent zur Variante ohne Pipe, mit dem Vorteil, dass das Resultat bei Bedarf über das Anhängen weiterer Pipes in weitere Funktionen geschoben werden kann.

```
el_tables <- html %>%
  html_nodes("table")
```

Im Objekt `el_tables` liegt nun eine Liste mit allen `<table>`-Elementen vor – der Name „el_tables" ist selbst gewählt. Interessieren nicht alle Tabellen, kann man den entsprechenden Listeneintrag über eckige Klammern auswählen, die vierte Tabelle enthält die Angaben zu deutschen Zeitschriften. Im nächsten Schritt gelangt man wieder über `html_nodes()` an die Zeilen der Tabelle:

```
el_rows <- el_tables[4] %>%
  html_nodes("tr")
```

Aus den einzelnen Zeilen können nun in einem Loop die Werte ausgelesen werden. Die Funktion `html_text()` erlaubt dabei Zugriff auf den Textinhalt der Elemente, während mit `html_attr()` einzelne Attribute extrahiert werden. So

---

[13] Die Funktion unterstützt auch XPath, siehe die Hilfe. Die Hilfe erreichen Sie in RStudio, nachdem der Cursor auf der Funktion platziert wurde, mit der Taste F1.

[14] In komplexeren Fällen – beispielsweise wenn es verschiedene Arten von Tabellen gibt – ist es häufig notwendig, dass man in den Selektoren zusätzlich Angaben zu Klassen oder IDs macht, um die gewünschten Elemente anzusteuern. Dafür wird die Bezeichnung der Klasse mit einem Punkt an das Element angehängt, die Bezeichnung der ID mit einem Doppelkreuz `#`. So könnte man beispielsweise die Tabelle mit der `id="123"` über den CSS-Selektor `table#123` ansteuern.

lässt sich neben dem Link auch der Text aus dem Element `<a href="/wiki/ADAC_Motorwelt" title="ADAC Motorwelt">ADAC Motorwelt</a>` auslesen.

Um alle Werte der einzelnen Zeilen einzusammeln, wird zunächst ein leeres Tibble `results` angelegt. Innerhalb eines jeden Durchlaufs wird dann ein Tibble `magazine` mit nur einer einzigen Zeile erstellt, welches mit `bind_rows()` fortlaufend an `results` angehängt wird:

```
# Leeres Tibble zum Sammeln der Ergebnisse
results <- tibble()

# Alle Zeilen abarbeiten
for (el_row in el_rows) {

  # Alle Spalten innerhalb einer Zeile finden
  el_cols <- el_row %>%
    html_nodes("td")

  # Text aus der zweiten Spalte auslesen
  titel = el_cols[2] %>%
    html_text()

  # URL aus dem Attribut 'href' auslesen
  link = el_cols[2] %>%
    html_nodes("a") %>%
    html_attr('href')

  # In einem neuen Tibble ablegen...
  magazine <- tibble(
    'titel' = titel,
    'link' = link
  )

  # ... und zu den Ergebnissen hinzufügen
  results <-  bind_rows(results, magazine)

}
```

Versuchen Sie, noch weitere Spalten auszulesen und in der `results`-Liste abzulegen!

7.1.2 Automatisierung des Webbrowsers

Eine weitere Möglichkeit zur Datenerhebung im Web besteht darin, den Browser zu automatisieren. Der Browser wird dabei wie eine Marionette durch Code ferngesteuert. Das folgende Beispiel verwendet Python, um den Browser Firefox zu automatisieren. Zwischen dem Skript und dem Browser sitzen drei weitere Komponenten (Abb. 7.2). Der Browser wird durch den GeckoDriver um eine Programmierschnittstelle erweitert, auf die dann die Automatisierungssoftware Selenium (ThoughtWorks 2022) zugreift. Die Entwicklung des Skripts, über welches Selenium angesteuert wird, findet in der Entwicklungsumgebung Jupyter-Notebook statt. Diese Pipeline ist nur ein Beispiel. Es können auch andere Browser wie Chrome oder andere Programmiersprachen und Entwicklungsumgebungen eingesetzt werden.

Die wesentliche Komponente zur Automatisierung des Browsers ist Selenium. Diese Software wird unter anderem zur Entwicklung von Webseiten eingesetzt, um automatisiert neu entwickelte Funktionen zu testen. Es lässt sich damit aber auch sehr gut Webscraping betreiben. Die Installation für Python nimmt man am besten auf der Kommandozeile über den Paketmanager pip vor. Wenn Sie mit JupyterLab arbeiten, können Sie direkt von dort ein Terminal öffnen (FILE > NEW LAUNCHER):

```
pip install selenium
```

Natürlich benötigen Sie auch einen Browser, der ferngesteuert werden kann. Falls noch nicht vorhanden, installieren Sie Firefox.[15] Alternativ können Sie auch Google Chrome verwenden, müssen dann aber in den nächsten Schritten statt des

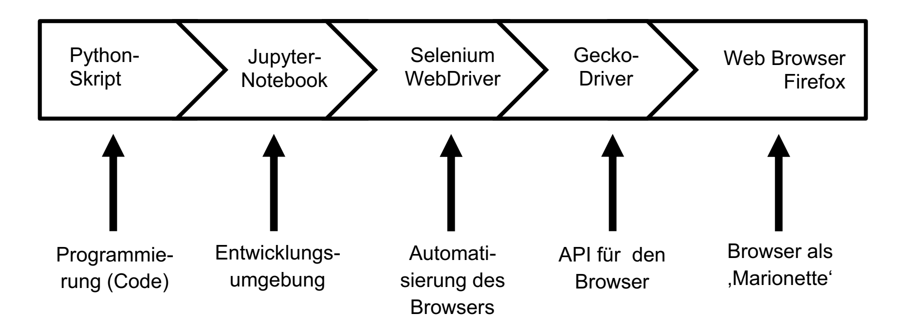


Abb. 7.2 Komponenten bei der Automatisierung eines Browsers am Beispiel von Firefox. (Quelle: eigene Darstellung)

[15] Siehe Mozilla (2022; https://www.mozilla.org/de/firefox/new/).

GeckoDriverManager den ChromeDriverManager einbinden. Diese Driver stellen die Schnittstelle zwischen Selenium und dem Browser dar, bei der Installation hilft der Webdriver Manager:[16]

```
pip install webdriver_manager
```

Starten Sie anschließend JupyterLab und legen Sie ein neues Python-3-Notebook an. Der passende Driver kann nun installiert und eingebunden werden:

```
from selenium.webdriver.firefox.service \
  import Service
from webdriver_manager.firefox \
  import GeckoDriverManager
driver = Service(GeckoDriverManager().install())
```

Sobald die Software installiert ist, sind die Schritte im Prinzip die gleichen wie beim klassischen Webscraping. Zunächst werden Seiten aufgerufen, dann Daten extrahiert und diese Daten werden dann abgespeichert.

Schritt 1: Browser fernsteuern Im ersten Schritt werden der Webbrowser gestartet und eine Seite geöffnet:

```
# Programmbibliotheken laden
from selenium.webdriver.firefox.service \
  import Service

from webdriver_manager.firefox \
  import GeckoDriverManager

# Driver initialisieren
driver = Service(GeckoDriverManager().install())

# Browser starten
browser = webdriver.Firefox(service=driver)
browser.get("https://www.google.de/")
```

Daraufhin sollte sich ein Firefox-Fenster geöffnet haben, in dem die Startseite von Google aufgerufen wird. Sie können nun manuell in diesem Browser surfen, das Surfen automatisieren oder zwischen diesen beiden Modi hin und her

[16] Zur Installation siehe Pirogov (2022; https://pypi.org/project/webdriver-manager/).

wechseln. So bietet es sich beispielsweise zu Beginn an, den Datenschutzbedingungen, die sich beim erstmaligen Aufrufen von Google öffnen, manuell zuzustimmen und erst dann die weitere Suche zu automatisieren.

Um automatisiert Suchbegriffe in den Suchschlitz einzugeben, muss der Suchschlitz eindeutig identifiziert werden. Wie beim klassischen Webscraping schauen Sie dazu in den Quelltext der Seite, zum Beispiel über die Entwicklerkonsole. Klicken Sie mit der rechten Maustaste auf den Suchschlitz und wählen Sie ELEMENT UNTERSUCHEN. Es handelt sich bei diesem Eingabefeld um ein HTML-Element mit dem Namen `input`. Dieses Element kann auf verschiedene Weise identifiziert werden, beispielsweise durch die ID `lst-ib` oder den Namen `q` (Abb. 7.3).

Selenium stellt mehrere Funktionen bereit, um mit den Elementen einer Seite zu arbeiten – Sie sollten sich dazu unbedingt die Dokumentation ansehen.[17] Das folgende Beispiel verwendet die Funktion `find_element()` in Kombination mit `By.Name` zur Identifizierung des Suchschlitzes. Zur Simulation der Tastatureingabe wird die Funktion `send_keys()` verwendet. Vorher wird festgelegt, dass Selenium einige Zeit warten soll, falls das gesuchte Element nicht verfügbar ist. Das ist notwendig, weil das Laden der Seite einen Moment dauert:

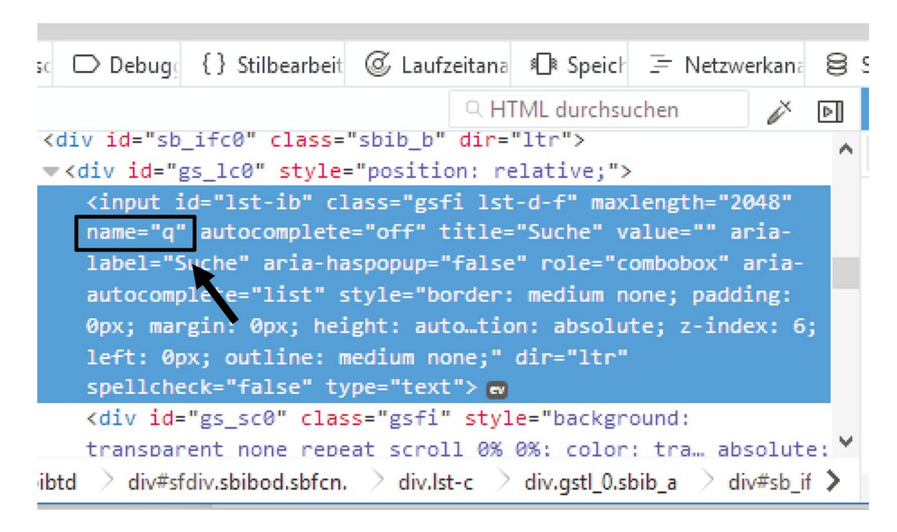


Abb. 7.3 Quelltext eines Eingabefelds. (Quelltext von https://www.google.de. Quelle: eigene Darstellung)

[17] Eine übersichtliche Einführung findet sich bei Muthukadan (2018; http://selenium-python.readthedocs.io/api.html).

```
# By.Name und By.Id importieren
from selenium.webdriver.common.by import By

# Beim Zugriff auf Elemente der Seite bis zu
# 10 Sekunden warten, sodass die Seite laden kann
browser.implicitly_wait(10)

# Suchschlitz finden
suchschlitz = browser.find_element(By.NAME, "q")

# In den Suchschlitz schreiben
suchschlitz.send_keys("Wie macht ein ")

# Abschicken
suchschlitz.submit()
```

Eingabefelder und Buttons sind auf Webseiten in der Regel in sogenannte Formulare eingebettet. Indem man das Formular abschickt, werden alle eingegebenen Inhalte an den Server geschickt. Die letzte Zeile schickt das Formular ab, das mit dem ausgewählten Eingabefeld in der Google-Suche verbunden ist. Dafür verfügt das `input`-Element (das im Objekt `suchschlitz` abgelegt ist) über die Funktion `submit()`. Alternativ zum Abschicken über `submit()`, welches dem Drücken der Enter-Taste gleichkommt, könnte auch der Klick auf den Button simuliert werden. In diesem Fall müssten Sie analog zum Suchschlitz den Suchbutton identifizieren. Für das gefundene Element könnte anschließend die `click()`-Funktion anstelle der `submit()`-Funktion ausgeführt werden.

Webscraping kann übrigens auch so durchgeführt werden, dass der Browser unsichtbar bleibt, das heißt, die Befehle können ausgeführt werden, ohne ein Browserfenster zu öffnen. Ein solcher Modus wird *headless* genannt und beschleunigt den Prozess in der Regel. Weitere Informationen dazu finden Sie in der Dokumentation von Selenium.

Schritt 2: Inhalte der Seite verarbeiten Auch um den Inhalt der Seite zu verarbeiten, müssen wieder die gewünschten Elemente über Namen oder IDs identifiziert werden. Dann kann auf den Inhalt der Elemente über die `text`-Eigenschaft zugegriffen werden. Im folgenden Beispiel wird außerdem gezeigt, wie aus diesem Text mit einem regulären Ausdruck (siehe Abschn. 4.1.1) die Zahl ausgelesen wird.

```
# Anzahl der Ergebnisse auslesen
ergebnisse = browser.find_element(
    By.ID, 'result-stats'
)

print(ergebnisse.text)

# Bibliothek für reguläre Ausdrücke
# zum Verarbeiten von Zeichenketten laden
import re

# Zahl mit regulärem Ausdruck extrahieren
anzahl = re.search(
    '([0-9\.]+) Ergebnisse',
     ergebnisse.text
).group(1)

# Punkt aus Zeichenkette entfernen
anzahl = anzahl.replace('.', '')

# In eine Ganzzahl (int) umwandeln
anzahl = int(anzahl)
print(anzahl)
```

Automatisierung wird dann sinnvoll, wenn mehrere Suchabfragen durchgeführt oder mehrere Seiten ausgelesen werden sollen. Die Ergebnisse können analog zur Vorgehensweise beim klassischen Webscraping in einer Liste bestehend aus Dictionaries abgelegt werden. Vorausgesetzt der Webbrowser wurde wie bislang beschrieben mit Selenium gestartet, lässt sich das gut über eine For-in-Schleife umsetzen:

```
# URL und Liste mit Keywords festlegen
url = "https://www.google.de/"
keywords = ["Computational", "Statistical", \
            "Interpretive"]

# Leere Liste für Suchergebnisse erstellen
results = []
```

```
# Für jedes Keyword die Google-Suche durchführen,
# Anzahl der Ergebnisse auslesen und
# in der results-Liste ablegen
for keyword in keywords:
    print(keyword)
    browser.get(url)

    suchschlitz = browser.find_element(By.NAME, "q")
    suchschlitz.send_keys(keyword)
    suchschlitz.submit()

    anzahl = browser.find_element(
        By.ID, 'result-stats'
    ).text

    result = {'keyword':keyword, 'count':anzahl}
    results.append(result)

results
```

Eine der größeren Herausforderungen besteht darin, festzustellen, wann eine Seite fertig geladen ist, sodass die benötigten Elemente zur Verfügung stehen. Eine grundsätzliche Einschränkung besteht auch darin, dass mit Selenium (bislang) keine http-Statuscodes ausgelesen werden können. So weiß man nicht, ob eine Seite überhaupt geladen wurde oder ob eine Fehlermeldung vorliegt. Deshalb sollte man Vorsichtsmaßnahmen einbauen.

Beim Zugriff auf einzelne Elemente wird dank der bislang verwendeten Anweisung `browser.implicitly_wait(10)` so lange gewartet, bis das Element geladen wurde. Das gilt zum Beispiel für die Funktion `find_element()`. Wenn das Element nach dem angegebenen Timeout von 10 Sekunden nicht gefunden wurde, wird das Skript abgebrochen. Schwierig wird diese Lösung allerdings immer dann, wenn das gesuchte Element auch schon auf der zuletzt aufgerufenen Seite vorhanden war. Denn in diesem Fall wird möglicherweise gar nicht der neue, sondern der alte Inhalt gefunden. Zur Lösung dieses Problem gibt es sehr verschiedene Ansätze. Die folgende Funktion fragt beispielsweise wiederholt den Status der Seite über JavaScript ab:

```
# Funktion definieren, um so lange zu warten,
# bis die Seite geladen ist
```

```
import time
def waitForReadyState(browser, timeout = 15):
    start_time = time.time()

    while time.time() < start_time + timeout:
        page_state = browser.execute_script(
            'return document.readyState;'
        )
        if page_state == 'complete':
            return True

        time.sleep(0.1)

    return False
```

Die Funktion könnte über `waitForReadyState(browser)` jedes Mal vor dem Zugriff auf ein Element aufgerufen werden, wobei der Parameter `browser` auf das vom Webdriver bereitgestellte Browser-Objekt verweist. Innerhalb der Funktion wird mittels `execute_script()` JavaScript im Kontext des Browsers ausgeführt. JavaScript wird häufig für die Interaktion auf Webseiten eingesetzt, hiermit lässt sich tief in die Funktionsweise einer Webseite eingreifen. Auch wenn Sie vielleicht im ersten Moment noch nicht alle Details verstehen, ist dies ein erster Ansatz, falls Sie selbst bei Ihren Experimenten auf das Problem stoßen, warten zu müssen. Weitere Techniken zur Überprüfung des Ladestatus bestehen darin, den alten und den neuen Seiteninhalt zu vergleichen oder so lange zu warten, bis ein Element in der alten Seite ungültig (engl. *stale*) wird – bei der Umsetzung von Webscraping ist also etwas Kreativität gefragt.[18]

Schritt 3: Daten abspeichern Wie schon im Abschnitt zum klassischen Webscraping beschrieben, können die Ergebnisse mit *pandas* wieder in einen Dataframe umgewandelt und als CSV-Datei abgespeichert werden.

```
# Bibliotheken einbinden
import pandas as pd
```

[18] Weitere Lösungen finden sich insbesondere auf Stack Overflow, etwa bei Apogne (2014; https://stackoverflow.com/questions/26566799/how-to-wait-until-the-page-is-loaded-with-selenium-for-python).

```
# Liste mit Dictionaries in DataFrame umwandeln
results = pd.DataFrame(results)

# DataFrame als CSV-Datei abpseichern
results.to_csv('results.csv', sep = ";", \
               index = False)
```

Mitunter lohnt es sich, den gesamten Quelltext der Seite für spätere Analysen abzuspeichern. Dieser Quelltext unterscheidet sich in einem entscheidenden Punkt von der HTML-Datei, die beim klassischen Webscraping heruntergeladen wird. Es handelt sich hier um den *generierten* Quelltext einer Seite, der durch Eingaben in Formulare oder durch interaktive Skripte (JavaScript) verändert werden kann. So kommt es beispielsweise vor, dass Kommentare auf Zeitungsseiten direkt in den HTML-Text eingebunden werden, die Antworten zu den Kommentaren allerdings nachgeladen werden, sobald die Leser:innen auf einen Button wie *Antworten zu diesem Kommentar anzeigen* klicken. Diese Antworten werden also interaktiv geladen und lassen sich nicht bereits über das bloße HTML-Grundgerüst erheben. Der generierte Quelltext entspricht ganz genau dem sichtbaren Inhalt der Seite, ein Vorteil der Automatisierung über Selenium. Dieser Quelltext wird über das Attribut `page_source` des Webdriver-Objekts bereitgestellt:

```
html = browser.page_source
```

Alternativ kann auch der Quelltext innerhalb einzelner Elemente ausgelesen werden. Selenium stellt dafür innerhalb des Document Object Model (DOM) eine Zugriffsmöglichkeit auf die `innerHTML`-Eigenschaft von Elementen bereit:

```
body = browser.find_element(By.TAG_NAME, 'body')
html = str(body.get_attribute('innerHTML'))
```

Um den Quelltext schließlich abzuspeichern, wird eine Datei zum Schreiben (Parameter `"w"`) geöffnet:

```
with open(
    "meineseite.html", "w", encoding="utf-8"
) as file:
    file.write(html)
```

Da jede Webseite anders aufgebaut ist, muss man sich für die Extraktion der Daten in die Struktur einer Seite eindenken. Will man viele verschiedene Seiten erheben, dann braucht man dagegen eine allgemeinere Technik, um irrelevante Teile wie das Menü, Werbung oder den Footer einer Seite zu entfernen. Für dieses sogenannte Boilerplate Removal gibt es viele verschiedene Ansätze und auch spezialisierte Python-Packages wie *trafilatura* (Barbaresi 2022; ☛ *Repositorium*).

Zur Dokumentation der Datenerhebung lassen sich darüber hinaus Screenshots der Seite automatisch abspeichern, wobei nur der sichtbare Bereich erfasst wird:

```
browser.save_screenshot('meineseite.png')
```

Mit Firefox können außerdem ganze Seiten oder ausgewählte Teile einer Seite als Screenshot gespeichert werden. Dazu wird zunächst das gewünschte Element identifiziert, zum Beispiel das `<body>`-Element, um die gesamte Seite zu erfassen. Da Screenshots Pixeldaten in Binärform enthalten, muss die Datei mit dem Parameter `"wb"` (= write binary) geöffnet werden:

```
body_element = browser.find_element(\
               By.TAG_NAME, 'body')
body_png = body_element.screenshot_as_png

with open("meineseite.png", "wb") as file:
    file.write(body_png)
```

Webscraping über die Automatisierung des Browsers besticht dadurch, dass hier das Surfen einer Nutzerin oder eines Nutzers simuliert wird. Man kann sich automatisiert bei Seiten anmelden sowie manuelles und automatisiertes Surfen kombinieren. Die Seiten sehen nicht anders aus als bei der manuellen Internetnutzung. Im Gegensatz zum klassischen Webscraping ist jedoch insgesamt mehr Geduld erforderlich und auch die Ersteinrichtung kann einige Zeit in Anspruch nehmen.

Alternative: Selenium mit R Auch mit R und RStudio können Browser über Selenium ferngesteuert werden (☛ *Repositorium*). Dazu installieren Sie das Package *RSelenium* (Harrison und Kim 2020). Wenn Sie es anschließend laden, werden bei der ersten Verwendung alle weiteren benötigten Komponenten automatisch nachinstalliert:

```
library(RSelenium)
```

Nach dem Laden der Bibliothek werden ein Selenium-Server und ein ferngesteuerter Browser gestartet:

```
selenium <- rsDriver(browser = "firefox",
                     port = 4566L)
server <- selenium$server
browser <- selenium$client
```

Analog zu Python, können nun Seiten aufgerufen werden:

```
browser$navigate("http://www.google.com")
```

Der Zugriff auf Eingabefelder gestaltet sich ebenfalls ähnlich:

```
suche <- browser$findElement(using = 'name', 'q')
suche$sendKeysToElement(list("Wie macht ein "))
suche$submitElement()
```

Die möglichen Befehle und Parameter finden Sie in der Hilfe des Packages *RSelenium*. Die Befehle zur Interaktion mit dem Browser sind unter „remoteDriver-Class" dokumentiert, die Befehle zum Auffinden, Auslesen und Interagieren mit den Elementen einer Seite unter „webElement-class". Am Ende sollten der Browser und der Server wieder geschlossen werden:

```
browser$close()
server$stop()
```

7.1.3 Spezialisierte Programme und Plattformen

Um die Komplexität von Webscraping zu bewältigen und den Programmieraufwand zu reduzieren, stehen mittlerweile vielfältige Programme zur Verfügung (siehe Tab. 7.1). Diese lassen sich danach einteilen, wie stark die notwendigen Skripte selbst entwickelt werden müssen oder wie sehr auf die Leistung von Dritten zurückgegriffen wird:

- Mit **Programmierbibliotheken** wie Scrapy für Python werden eine Vielzahl von Funktionen zum Webscraping bereitgestellt und einige der gängigen Herausforderungen umschifft. Für andere Programmiersprachen gibt es ebenfalls hilfreiche Bibliotheken, etwa Apify für JavaScript.

- Mit **lokalen Programmen** wie HTTrack oder Facepager kann der erste Schritt, das Herunterladen der Quelltexte, umgesetzt werden. Gerade für groß angelegte Datenerhebungen ist dies eine Erleichterung, da Seiten auch parallel heruntergeladen werden können. Programme wie ScrapeBox oder Facepager lesen darüber hinaus gezielt Links aus und können damit als Webcrawler eingesetzt werden. Wenn umfangreiche Funktionen zum Extrahieren und Analysieren der Daten benötigt werden, ist beispielsweise RapidMiner eine Option.
- Die bislang genannten Programme werden einmal installiert und laufen dann auf dem lokalen Computer. Bei **webbasierten Plattformen** wie ScrapeBot findet die Datenerhebung und -analyse dagegen auf einem Server statt. Nicht immer ist sofort erkennbar, dass es sich um einen serverbasierten Dienst handelt. So wird das Chrome Plugin von Webscraper.io beispielsweise lokal im Browser installiert, dennoch findet die Datenaufbereitung auf dem Server statt. Von Vorteil ist in der Regel die hohe Performanz serverbasierter Dienste, bedingt durch eine gute Internetanbindung und parallelisierte Prozesse. Neben kostenlosen Einstiegstarifen werden für höhere Leistungsanforderungen mitunter Entgelte erhoben. Auch die Bedienung ist bei kommerziellen Anbietern durchaus komfortabel, denn viele technische Details verschwinden hinter gefälligen Oberflächen. Allerdings gibt man dabei insofern die Hoheit über den Erhebungsprozess auf, als dass die konkrete Umsetzung nicht mehr nachvollzogen werden kann und auch die Daten beim jeweiligen Anbieter gespeichert werden. Diese Vorgehensweise sollte im wissenschaftlichen Kontext gut überlegt sein. Auch im akademischen Umfeld werden Webcrawler angeboten. Eine vielfältige Sammlung spezialisierter Scraper bietet etwa die Digital Methods Initiative (DMI) aus Amsterdam.

Tab. 7.1 Beispiele für Programmbibliotheken, Programme und Cloud-Dienste zum Webscraping

Programm	Funktionen
Bibliotheken und Frameworks	
Apify SDK	JavaScript/Node.js-Bibliothek für Webscraping (Apify Technologies 2022; https://sdk.apify.com)
rvest und httr	R-Pakete für Webscraping (Wickham 2022b; https://github.com/tidyverse/rvest) (Wickham 2022a; https://github.com/r-lib/httr)
Scrapy	Python-Bibliothek für Webscraping (Zyte 2022; https://scrapy.org)
Apache Nutch	Tool für die Kommandozeile (Apache Software Foundation 2021; https://nutch.apache.org/)

(Fortsetzung)

Tab. 7.1 (Fortsetzung)

Programm	Funktionen
Desktop-Programme	
Facepager	Automatisiertes Herunterladen von Dateien, Webscraping und Extrahieren von Links (Jünger und Keyling 2022; https://github.com/strohne/Facepager)
HTTrack Website Copier	Lokale Spiegelung einer Webseite (Roche 2017; https://www.httrack.com/)
RapidMiner	Kommerzielles Analyseprogramm mit umfangreichem Funktionsumfang unter anderem für Webcrawling, Scraping und Analyse (RapidMiner 2021; https://rapidminer.com)
SocSciBot	Web Crawler zur Erhebung und Analyse von Links und Texten (Statistical Cybermetrics Research Group 2016; http://socscibot.wlv.ac.uk)
ScrapeBox	Automatisiertes Herunterladen von Dateien und Extraktion von Links (ScrapeBox 2022; http://www.scrapebox.com)
Serverbasierte Anwendungen	
DMI Tools	Spezialisierte Webscraper der Digitial Methods Initiative, das heißt aus dem akademischen Kontext (Helmond 2020; https://wiki.digitalmethods.net/Dmi/ToolDatabase)
Octoparse	Kommerzieller Webscraping-Dienst mit umfangreichen Funktionen (Octopus Data 2022; https://www.octoparse.com)
ScrapeBot	Sogenanntes agentenbasiertes Testen, bei dem Seiten regelmäßig überprüft werden (Haim 2021; https://github.com/MarHai/ScrapeBot)
Webscraper.io	Erweiterung für den Browser Chrome, wobei das Webscraping auf dem Server des Anbieters stattfindet (Web Scraper 2021; https://webscraper.io)

Anmerkung: Beachten Sie, dass Dienste kommen und gehen. Insbesondere im wissenschaftlichen Kontext sollte der Einsatz kommerzieller Dienste gut überlegt sein. Quelle: Eigene Darstellung

Auch wenn die Arbeit mit spezialisierten Programmen und Plattformen effizient ist, sind ein Grundverständnis von Webscraping und eigene Erfahrungen mit der Entwicklung von Webscrapern aus wissenschaftlicher Sicht wichtig. Denn dieses Wissen hilft nicht nur bei der Formulierung der Anforderungen, sondern vor allem bei der Einschätzung der Datenqualität und von typischen Fehlerquellen.

Zudem müssen rechtliche und ethische Aspekte reflektiert werden, vor allem das Urheberrecht und der Datenschutz. Webscraping ist tendenziell eine unhöfliche Art der Datenerhebung, da es die Server belastet. Einige Anbieter schließen Webscraping deshalb explizit in ihren Nutzungsbedingungen aus und ergreifen technische Maßnahmen zur Verhinderung von Webscraping. Hinweise zur Ethik finden Sie beispielsweise bei Thelwall und Stuart (2006) und eine rechtliche Beurteilung bei RatSWD (2019).

Übungsfragen

1. Fassen Sie kurz zusammen, welche Schritte beim Webscraping vollzogen werden müssen!
2. Worin unterscheiden sich klassisches Webscraping und Browserautomatisierung?
3. Wie unterscheidet sich der heruntergeladene Quelltext einer Seite vom generierten Quelltext?
4. Wie werden beim Webscraping die einzelnen Elemente einer Webseite identifiziert? Öffnen Sie eine Seite Ihrer Wahl, wählen Sie darauf ein Element aus und überlegen Sie sich eine Möglichkeit, dieses Element zu adressieren!
5. Mit welcher Python-Bibliothek können HTML-Texte geparst werden, um Daten zu extrahieren?
6. Was ist mit Boilerplate Removal gemeint?

7.2 Application Programming Interfaces (APIs)

Application Programming Interfaces (APIs) sind Programmierschnittstellen, über die festgelegt wird, wie zwei Programme miteinander interagieren können (Jacobson et al. 2012, S. 5). Will beispielsweise ein Immobilienportal seine Häuser und Wohnungen auf einer Karte anzeigen und eine Suche nach Orten umsetzen, kann es dafür die Karte von Google Maps über die Google Maps API einbinden. Diese Schnittstellen können auch Wissenschaftler:innen nutzen, um Daten abzufragen oder zu analysieren. Dazu werden meistens sogenannte REST-APIs[19] verwendet, die ihre Dienste über das Web anbieten. Nicht immer sind mit APIs ausschließlich solche webbasierten Dienste gemeint, das vorliegende Kapitel beschränkt sich aber auf REST-APIs, da sie sich gut für den Einstieg in die Welt der automatisierten Datenerhebung eignen. Die Vorgehensweise ist dabei immer gleich: Zunächst wird eine Anfrage an eine API gestellt. Zum Beispiel wird ein Ortsname an den Google-Maps-Server geschickt (Input). Die API verarbeitet die Anfrage und liefert ein Ergebnis zurück, zum Beispiel die Geokoordinaten des Ortes (Output). Diese Koordinaten können dann verwendet werden, um passende

[19] REST steht für Representational State Transfer und bezeichnet unter anderem das Grundprinzip des Web: einzelne Ressourcen lassen sich über festgelegte URLs ansprechen (Fielding 2000). Anwendungen wie etwa Messengerdienste verwenden teilweise eigene API-Protokolle, beispielsweise verwendet Telegram das speziell für die eigenen Zwecke entworfene MTProto Mobile Protocol (Telegram 2020).

Immobilien anzuzeigen. Wie webbasierte APIs grundsätzlich funktionieren, wie darüber Daten erhoben werden können und wie die Schnittstellen auch anderweitig für die Datenanalyse verwendet werden können, wird in diesem Kapitel theoretisch und praktisch beantwortet.

Warum lohnt sich eine Auseinandersetzung mit APIs? Der Einstieg in eine API kann durchaus Zeit in Anspruch nehmen, wenn man sich erst bei einem Anbieter registrieren, die Dokumentation verstehen und passende Skripte entwickeln muss. Dann aber kommen die Vorteile von APIs gegenüber anderen Datenzugängen zum Tragen. Sie liefern einen über längere Zeiträume stabilen Zugang, der in der Regel gut dokumentiert und nach der Ersteinrichtung einfach zu handhaben ist. Die Daten sind vorstrukturiert, zum Beispiel im JSON-Format, sodass sich der Aufwand bei der Datenaufbereitung in Grenzen hält. Schließlich lassen sich darüber auch komplexe Analysen wie die automatisierte Bilderkennung oder die Transkription von Audiodateien vergleichsweise einfach umsetzen. Dabei ist allerdings immer zu beachten, dass die Analyse über einen Dienst läuft, dessen interne Mechanismen meist aus Sicht der Wissenschaftler:innen eine *black box* sind. Insofern gilt es stets, die Datenqualität einzuschätzen und zu überprüfen.

7.2.1 Input: Anfragen an eine API

Da jede API anders funktioniert, lässt sich keine allgemeingültige Anleitung für APIs schreiben. Stattdessen sind alle notwendigen Informationen in den Dokumentationen der jeweiligen API festgehalten. Diese stellen ein grundlegendes Handbuch zur API dar und sind in den Entwicklerportalen oder über Suchmaschinen unter Stichworten wie „Reference“ zu finden. Die Dokumentationen beinhalten üblicherweise eine Übersicht über die verschiedenen APIs eines Dienstes – Anbieter schnüren unterschiedliche Pakete wie eine kostenfreie API oder eine API für Unternehmen. Dort finden sich auch Getting-Started-Anleitungen oder weitere Einstiegshilfen. Besonders die APIs von großen Diensten können auf den ersten Blick sehr kompliziert wirken. Hier lohnt es sich, sich zunächst über die angebotenen Hilfestellungen mit der Struktur und den Möglichkeiten der API vertraut zu machen.

Die Dokumentation enthält insbesondere eine Auflistung der sogenannten Endpunkte. Endpunkte werden bei webbasierten APIs die URLs genannt, über die verschiedene Daten wie Geokoordinaten oder Posts abgefragt (oder auch erstellt) und über die Funktionen wie die Bild- oder Texterkennung ausgeführt werden können. In der Dokumentation wird angegeben, wie sich die URLs zusammensetzen, welche weiteren Parameter eine Abfrage benötigt und wie das Ergebnis aussieht. Bei der Interaktion mit einer API schickt ein Client – beispielsweise ein selbstge-

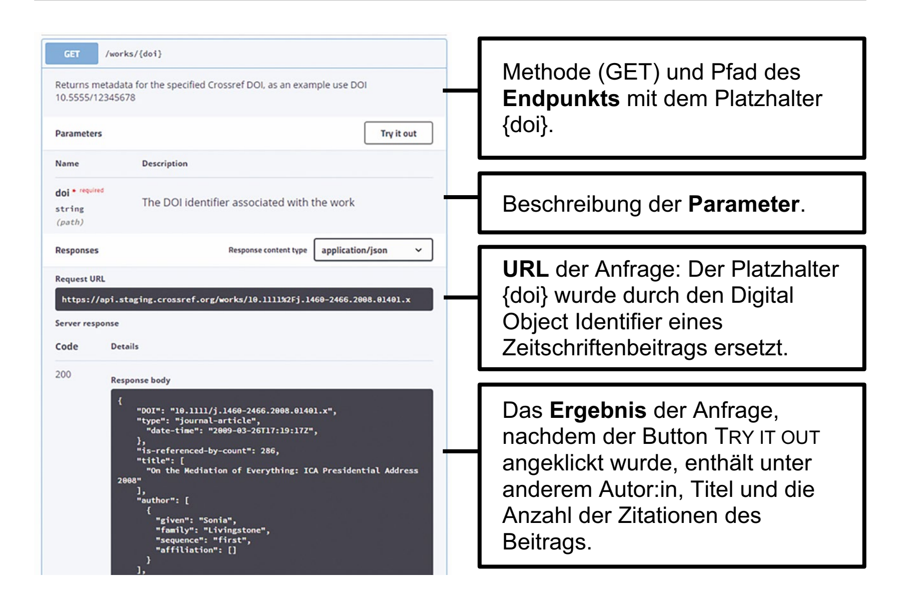


Abb. 7.4 Swagger User Interface für die CrossRef-API zur Abfrage von Daten über Zeitschriftenbeiträge (gekürzt). (Quelle: Crossref (2022; https://api.staging.crossref.org/swagger-ui/index.html))

schriebenes Programm – über diese URL eine Anfrage an den Server, bei dem die gewünschten Daten oder Funktionen vorhanden sind. Der Server verarbeitet diese Anfrage und gibt die entsprechenden Informationen an den Client zurück. Client und Server werden demnach über die API als Schnittstelle miteinander verbunden.

Einige Dienste stellen eine Webseite bereit, auf der die Endpunkte und verschiedene Parameter direkt ausprobiert werden können. Verbreitet ist dafür zum Beispiel Swagger UI (Abb. 7.4) oder auf Facebook wird dazu der Graph API Explorer angeboten.

Eine API-Anfrage muss unterschiedliche Elemente beinhalten, damit sie vom Server verstanden und akzeptiert wird: Neben der URL gehören dazu Anfrageheader, eine Methode und für einige Anfragen zudem eine Payload. Die **URL** setzt sich aus dem Protokoll, der Domain, dem Pfad und Parametern zusammen (ausführlich siehe Kap. 2). Eine URL ist dabei eine eindeutige Adresse, die zu einer Ressource im Web führt. Ressourcen sind Webseiten, die beispielsweise Tabellen, Profile, Posts oder Mediendateien enthalten. Achten Sie beim Surfen im Internet darauf, wie sich die URL verändert, wenn Sie Links und Buttons anklicken! Die URL einer API beginnt in der Regel immer gleich und enthält

mitunter die Version der API, zum Beispiel: `https://graph.facebook.com/v13.0`. An diesen Basispfad werden dann die Pfade der einzelnen Endpunkte angehängt, zum Beispiel ergibt sich daraus `https://graph.facebook.com/v13.0/{page-id}/posts`, um die Posts einer Facebook-Seite zu erhalten.

Parameter sind ein Teil der URL, wobei sich zwei Formen unterscheiden lassen. **Pfadparameter** sind Platzhalter im Pfad der URL, die durch eigene Werte ersetzt werden, etwa durch den Benutzernamen der Facebookseite des WWF, um die Posts dieser Seite zu erhalten: `https://graph.facebook.com/v13.0/wwfde/posts` (siehe auch Abb. 7.4 für ein weiteres Beispiel).

Query-Parameter sind dagegen Name-Wert-Paare, die mit einem ?-Zeichen an den Pfad angehängt werden. Mehrere Parameter werden mit einem &-Zeichen voneinander getrennt. So kann beispielsweise der Zeitraum von Posts eingegrenzt werden: …`/wwfde/posts?since=2021-03-01&until=2021-04-01`. Einige Parameter sind notwendig, so muss immer der Benutzername einer Facebook-Seite angegeben werden, um Posts dieser Seite zu erheben. Mit optionalen Parametern werden zusätzliche Einstellungen getätigt, beispielsweise die Eingrenzung der Zeiträume oder bei der Erhebung von Kommentaren, ob diese chronologisch oder nach Relevanz sortiert ausgegeben werden sollen.

Auch die Paginierung wird häufig über Query-Parameter umgesetzt. Wenn Sie mit Google etwas suchen, erhalten Sie in der Regel nicht alle Ergebnisse auf einmal zurück, sondern nur eine Seite mit den ersten Treffern, andernfalls wäre das Ergebnis kaum zu handhaben. Auch die Anfrage an eine API gibt in der Regel nicht alle Daten auf einmal zurück, sondern immer nur eine einzelne Seite. Das konkrete Vorgehen, um weitere Ergebnisse oder Seiten zu erhalten, unterscheidet sich von API zu API. Häufig kann die Anzahl der Ergebnisse auf einer einzelnen Seite über Parameter wie `limit=10` oder `limit=100` gesteuert werden und verschiedene Seiten werden über Parameter wie `page=1` oder `page=2` angefragt. Eine Alternative ist die cursor-basierte Paginierung: Bei jeder Anfrage wird dazu ein festgelegter Parameter aus der vorangegangenen Anfrage mitgeschickt (der Cursor), sodass die API erkennt, ab welchem Datensatz die Ergebnisliste fortgesetzt werden muss.

Bei jedem Aufruf einer URL werden zudem **Anfrage-Header** an den Server geschickt. Diese enthalten weitere Informationen zur Verarbeitung, sind im Gegensatz zu den Parametern aber nicht in der URL sichtbar. Die Header enthalten unter anderem Angaben zum verwendeten Client, dem sogenannten User Agent. Schauen Sie sich einmal in der Entwicklerkonsole eines Browsers an, welche Header beim Surfen im Web mitgeschickt werden.[20] Auch wenn eine API verschiedene Datenformate

---

[20] Dazu rufen Sie etwa unter Firefox die Entwicklerkonsole mit F12 auf, wechseln in den Reiter Netzwerkanalyse und klicken eine der Anfragen an.

wie JSON oder XML (siehe Kap. 3) unterstützt, kann ein Client das gewünschte Format mitunter über den Accept-Header angeben, beispielsweise mit `Accept: application/json`.

Die **Methode** der Anfrage teilt dem Server mit, welche Operation auf der angefragten Ressource ausgeführt werden soll. Sie wird über sogenannte HTTP-Verben angegeben, unter anderem:

- GET: Daten von einer Ressource werden gelesen bzw. abgefragt.
- POST: Daten werden zu einer Ressource auf einem Server geschickt, um sie dort zu speichern.
- DELETE: Die Ressource soll gelöscht werden.

Bei der Interaktion mit APIs sind im wissenschaftlichen Kontext besonders die Methoden GET und POST üblich. Mit POST werden umfangreichere Daten als sogenannte **Payload** übertragen – zum Beispiel, wenn Audiodateien oder Bilddateien bei einer Anfrage mitgeschickt werden sollen.

7.2.2 Output: Die Antwort einer API

Nicht jede Anfrage an einen Server ist erfolgreich: Wenn Zugriffsrechte fehlen, die URL oder andere Elemente der Anfrage falsch formatiert sind oder eine Ressource nicht (mehr) existiert, quittiert dies der Server mit einem entsprechenden Statuscode (Tab. 7.2). Im besten Fall finden sich in der Antwort des Servers weitere Informationen zur Fehlerquelle, etwa welcher Parameter der Anfrage nicht korrekt angegeben wurde. Die Antwort enthält auch Response-Header, in denen manchmal kurze Hinweise auf fehlende Zugriffsrechte erläutert werden.

Tab. 7.2 Typische Statuscodes beim Umgang mit APIs

Statuscode	Beschreibung
200 OK	Die Anfrage war erfolgreich.
400 Bad Request	Die Anfrage ist falsch formatiert.
403 Forbidden	Der Zugriff ist nicht erlaubt.
404 Not Found	Die Ressource existiert nicht.
429 Too Many Requests	Es wurden zu viele Anfragen gestellt (rate limit).
500 Internal Server Error	Auf dem Server ist ein Fehler aufgetreten.

Quelle: Eigene Darstellung auf Grundlage von Fielding et al. (2022)

```
{
  "id": "288837164503756",
  "name": "Greifswalder Institut für Politik-
  "location": {
    "city": "Greifswald",
    "country": "Germany",
    "latitude": 54.098440810237,
    "located_in": "234989956684237",
    "longitude": 13.375945687294,
    "street": "Ernst-Lohmeyer-Platz 3",
    "zip": "17489"
  },
  "about": "Greifswalder Institut für Politik
  "description": "Das Greifswalder Institut f
Kommunikationswissenschaft – kurz IPK – ist e
Instituts für Politikwissenschaft und des zuv
angesiedelten Lehrstuhls für Kommunikationswi
ist das bislang einzige seiner Art in Deutsch
  "fan_count": 727,
```

Abb. 7.5 Inhalt einer Facebook-Seite im Browser und als JSON-Format. (Quelle: eigene Darstellung)

War die Anfrage erfolgreich, werden die angefragten Daten im Response Body zurückgegeben und können auf dem Client abgespeichert werden. Ein typisches Datenformat dafür ist JSON (ausführlich siehe Abschn. 3.5). Im JSON-Format werden Daten als Objekte in geschweiften Klammern `{}` zurückgegeben. Objekte enthalten wiederum Schlüssel-Wert-Paare: Die Bezeichnung eines Feldes wird in Anführungszeichen angegeben, es folgen ein Doppelpunkt und danach der Inhalt des Feldes. Mehrere Objekte in einer Liste werden in eckigen Klammern `[]` zusammengefasst.

Abb. 7.5 zeigt die Profilinformationen einer Facebook-Seite, wie sie im Browser angezeigt und von einer API als JSON-Objekt zurückgegeben werden. Praktisch an API-Daten ist, dass diese immer identisch strukturiert sind, egal welche Facebook-Seite abgefragt wird. Auch wenn andere Facebook-Seiten abgefragt werden, lässt sich über den Schlüssel `description` immer die Seitenbeschreibung und über den Schlüssel `location` immer die hinterlegte Ortsangabe der jeweiligen Facebook-Seite auslesen. Finden Sie im Beispiel, über welchen Schlüssel die Anzahl der Personen, denen die Seite gefällt, angegeben wird? Durch diese Strukturierung lassen sich Daten leicht in ein Tabellen-Format überführen, das sich wiederum gut für weitere Analysen eignet (Abb. 7.6). Dabei werden die Schlüssel als Spalten verwendet und jedes Objekt wird in einer eigenen Zeile abgebildet. Die Werte wiederum sind in den entsprechenden Zellen zu finden.

JSON-Format


```
{"data": [
  {
    "id": "145718628797076",
    "name": "webmoritz.de",
    "fan_count": 1470
  },
  {
    "id": "340018314585",
    "name": "Greifswald ...",
    "fan_count": 2732
  },
  {
    "id": "182566151762895",
    "name": "Fennistik ...",
    "fan_count": 1051
  }
]}
```


Tabelle

id	name	fan_count
14571…	webmoritz.de	1470
34001…	Greifswald …	2732
18256…	Fennistik …	1051

Abb. 7.6 Likes einer Facebook-Seite im JSON-Format und als Tabelle. (Quelle: eigene Darstellung)

7.2.3 Zugang zu APIs: Authentifizierung und Rate Limits

APIs sind häufig zugangsbeschränkt, zunächst muss man sich deshalb bei einem Dienst registrieren und für die API freischalten lassen. Dazu stellen Social-Media-Plattformen wie Facebook oder Twitter und auch Cloud-Computing-Dienstleister wie Google oder Amazon eigene Portale für Entwickler:innen zur Verfügung. Hier registriert man eine App oder ein Projekt, womit aber keine tatsächlich fertig programmierte Anwendung gemeint ist, sondern lediglich ein Konto bei dem jeweiligen Dienst. Über diese App erhält man Zugangsdaten zur API, wobei insbesondere zwei Verfahren verbreitet sind:

- Im einfachsten Fall erhält man direkt ein **Access Token**, das wie ein Passwort funktioniert und bei Anfragen an die API als Parameter an die URL angehängt oder im Header der Anfrage (siehe oben) mitgeschickt wird.
- Viele Anbieter setzen auf das **OAuth-Protokoll** (Open Authorization), das Login-Prozeduren für ganz unterschiedliche Szenarien beinhaltet. Wenn Sie sich beispielsweise mit einem Facebook-Konto auf der Seite eines Onlineshops anmelden können, wird dies über OAuth abgewickelt. Hier kommen drei Parteien zusammen: Facebook vermittelt an den Shop im Namen der Nutzer:innen den Zugriff auf die bei Facebook hinterlegten Daten, zum Beispiel die E-Mailadresse, sodass der Shop-Betreiber keine eigene Registrierung anbieten muss. Um eine solche Schnittstelle zu benutzen, erhält man vom API-Betreiber eine Client ID

und ein Client Secret. Mit diesen Daten kann dann ein Access Token erzeugt werden, das wiederum in jeder Anfrage an die API mitgeschickt wird.

Wie die Registrierung funktioniert, ist üblicherweise auf der Webseite der jeweiligen API dokumentiert.[21] Dort finden sich auch Angaben dazu, welche Daten überhaupt abgefragt werden können. Eine wichtige Limitation sind die sogenannten Rate Limits. Damit begrenzt beispielsweise Twitter, wie schnell die Follower eines Twitterprofils abgefragt werden können: In der Standard-API ist dies aktuell auf 15 Abfragen innerhalb eines Zeitfensters von 15 Minuten begrenzt. Will man direkt die Namen der Follower erheben (GET followers/lists), so werden in einer Abfrage bis zu 200 Follower zurückgegeben. Ein anderer Endpunkt (GET followers/ids) liefert pro Abfrage die IDs von bis zu 5000 Followern, allerdings ohne Namen und weitere Profilinformationen.[22] Gerade bei größeren Projekten, in denen man viele Daten über eine API erheben will, sollte man sich deshalb mit den Ratenbegrenzungen und möglichen Kombinationen von Abfragen auseinandersetzen.

Teilweise können höhere Rate Limits auch über Zugänge zu einer Premium API oder Enterprise API gekauft werden, hier finden sich sowohl pauschale Preismodelle als auch Modelle, in denen einzelne Abfragen abgerechnet werden. Google erlaubt beispielsweise aktuell für die automatisierte Bilderkennung pro Monat 1000 kostenlose Abfragen, für jedes weitere Paket von 1000 Anfragen werden anschließend 1,50 Dollar berechnet.[23] Einige Anbieter wie Twitter und Facebook stellen mittlerweile für wissenschaftliche Forschungsprojekte kostenlose oder erweiterte API-Zugänge zur Verfügung. Bei der Registrierung müssen dazu allerdings Begutachtungsverfahren durchlaufen werden – die Anbieter bestimmen selbst, welche Projekte sie genehmigen und welche nicht.

7.2.4 Tools für die Datenerhebung mit APIs

Ist der Zugang zur API geklärt, ist die Arbeit mit einer API trotz der vielen bislang besprochenen Details im Wesentlichen einfach und unterscheidet sich kaum vom normalen Surfen im Web: Eine URL wird zusammengesetzt, aufgerufen und das

[21] Zum Beispiel unter Twitter (2022c; https://developer.twitter.com/en/docs/platform-overview).

[22] Dokumentiert bei Twitter (2022f; https://developer.twitter.com/en/docs/twitter-api/v1/accounts-and-users/follow-search-get-users/api-reference/get-followers-list) und Twitter (2022a; https://developer.twitter.com/en/docs/twitter-api/v1/accounts-and-users/follow-search-get-users/api-reference/get-followers-ids).

[23] Siehe Google (2022e; https://cloud.google.com/vision/pricing).

Ergebnis wird abgespeichert. Sie können URLs für API-Anfragen also durchaus manuell zusammenstellen (sofern keine Authentifizierung nötig ist). Wenn Sie diese URLs in den Browser eintippen, sollten Sie die Daten der Abfrage angezeigt bekommen und können diese herunterladen, abspeichern und anschließend aufbereiten und auswerten.

Zum Ausprobieren oder für eine einmalige Anfrage mag die manuelle Vorgehensweise hilfreich sein. Soll eine API aber automatisiert genutzt werden, gibt es weitere Möglichkeiten, an denen Sie ansetzen können:

- Mit dem Tool cURL (Stenberg 2022) können Sie die Anfragen über die **Kommandozeile** formulieren. In **R** helfen besonders die Pakete *httr* zum Senden von Anfragen (Wickham 2022a) und *jsonlite* (Ooms 2014), um die heruntergeladenen Daten in lesbare Datensätze umzuwandeln. Entsprechende **Python**-Packages sind *requests* (Reitz 2022) und *json* (Python Software Foundation 2022c).
- Für einige APIs finden sich fertige **Packages** für Python oder R, die speziell auf eine bestimmte API zugeschnitten sind und eine Vielzahl von gut dokumentierten Funktionen implementieren. Um beispielsweise auf die Twitter-API zuzugreifen, lassen sich in R die Packages *rtweet* (Kearney 2019) oder *twitterR* (Gentry 2015) bzw. in Python das Package *Tweepy* (Roesslein 2020) verwenden.
- Anbieter von Social-Media-Plattformen oder Cloud-Computing-Diensten stellen sogenannte **Software Development Kits (SDKs)** für verschiedene Programmiersprachen wie JavaScript, PHP, Java oder Python sowie für Betriebssysteme wie Android oder iOS bereit. Damit lassen sich zum Beispiel auf diese Plattformen zugeschnittene Smartphone-Apps (native Anwendungen) bauen.

Darüber hinaus gibt es zahlreiche kommerzielle Dienste oder Tools mit Benutzeroberflächen, über die APIs bedient werden können. Wenn Sie bislang keine Programmierkenntnisse haben und einen schnellen Einstieg in die Datenerhebung über APIs erleben wollen, sind Tools mit Benutzeroberfläche eine gute Option. Darüber können Sie Abfragen relativ einfach zusammenklicken. Eine Vielzahl von Prozessen läuft dabei automatisch im Hintergrund ab und wird von den Programmen für Sie erledigt. Sie sollten sich nach und nach ein Grundverständnis dieser Prozesse erarbeiten. Als universelle Lösungen können Sie zum Beispiel das speziell für den Einstieg in die automatisierte Datenerhebung entworfene Facepager[24]oder das bei Programmierer:innen beliebte Postman[25] einsetzen. In den folgenden Kapi-

[24] Siehe Jünger und Keyling (2022; https://github.com/strohne/Facepager).

[25] Siehe Postman (2022; https://www.postman.com).

teln werden einmal mit Facepager die Erhebung von Twitter-Daten und einmal mittels des R-Package *googleCloudVisionR* (Pal et al. 2020) die automatisierte Bilderkennung über Google illustriert.

7.2.5 Social-Media-Daten über Facepager erheben: Followees eines Twitter-Accounts

Facepager ist ein Open-Source-Tool für die automatisierte Datenerhebung mittels APIs und Webscraping. Das Programm ist besonders für Anfänger lohnend, da keine Programmierkenntnisse notwendig sind und weil es einige Einstiegshilfen bereitstellt. Vor allem eine Erhebung von Daten über die APIs von Facebook, YouTube und Twitter ist über die dafür zugeschnittenen Module unkompliziert, sofern man Zugang zur API der jeweiligen Plattform hat. Das folgende Beispiel verdeutlicht das Prinzip, das sich dann auf andere APIs übertragen lässt. Weitere Möglichkeiten sind im ☛ *Repositorium* des Buchs aufgeführt.[26] Weitere APIs können über das universellere Generic Modul abgefragt werden. Für den allerersten Einstieg sind die Getting-Started-Anleitungen im Wiki von Facepager zu empfehlen.[27] Dort finden sich auch Hinweise zur Auswertung der erhobenen Daten oder ausführliche Erklärungen zu den einzelnen Einstellungen von Facepager. Im zweiten Schritt helfen die Presets weiter, um für einige Szenarien die nötigen Voreinstellungen zu übernehmen, darin sind ebenfalls Erklärungen der notwendigen Schritte zu finden.

Das Layout von Facepager ist in Abb. 7.7 zu sehen. In der Menü-Leiste sind einige der wichtigsten Funktionen aufgeführt, vor allem das Erstellen neuer Datenbanken oder das Hinzufügen von neuen Knoten (engl. *nodes*) – so werden in Facepager die einzelnen Datensätze wie etwa Posts oder Kommentare bezeichnet. Über das Query Setup können die verschiedenen Module ausgewählt und alle notwendigen Einstellungen für die Abfrage vorgenommen werden. Im Nodes View sind alle Datenknoten tabellarisch aufgelistet, wobei jeder Knoten eine Object ID enthält. Wenn man einen Knoten anklickt, sieht man im Data View die gesamten erhobenen Daten zu einem Knoten. Über das Column Setup kann festgelegt werden, welche dieser Daten in eigenen Spalten angezeigt und exportiert werden sollen. Das Status Log zeigt während der Erhebung den aktuellen Fortschritt und Fehlermeldungen an.

Wie diese einzelnen Elemente zusammenspielen, wird nachfolgend anhand eines Beispiels beschrieben, bei dem die Followees von Twitter-Profilen (Profile,

[26] Mit Facepager können aktuell vorregistrierte Zugänge benutzt werden, womit die Registrierung einer eigenen App bei den API-Anbietern entfällt. Inwiefern Facepager diesen Service in Zukunft weiter anbieten kann, hängt von Entwicklungen bei den Onlineplattformen ab.

[27] Siehe Jünger (2020; https://github.com/strohne/Facepager/wiki).

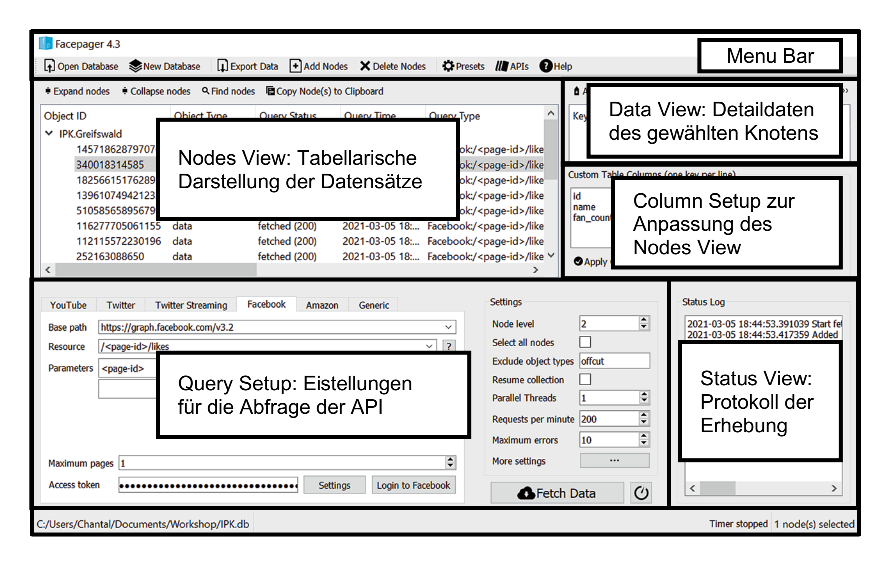


Abb. 7.7 Layout von Facepager. (Quelle: eigene Darstellung)

denen ein Profil folgt) abgefragt werden. Eine solche Erhebung kann Grundlage für Netzwerkanalysen sein, um die Verbindungen zwischen verschiedenen Organisationen oder die Diskursgemeinschaften zu aktuellen Themen sichtbar zu machen (siehe Kap. 10). Allerdings: Auch wenn APIs relativ stabil sind und das Programm fortlaufend an aktuelle Entwicklungen in der Online-Welt angepasst wird, wenn Sie das hier lesen, kann sich die Situation schon wieder geändert haben und möglicherweise sind einige Plattformen nicht mehr frei Haus zugänglich oder die unten beschriebenen Mausklicks funktionieren nicht mehr auf die gleiche Weise. Im Zweifelsfall lohnt es sich immer, die aktuellen Dokumentationen der APIs oder verwendeten Tools direkt anzusehen – diese werden üblicherweise von den Betreibern auf dem aktuellen Stand gehalten.

Die grundsätzliche Vorgehensweise bleibt immer gleich: Zunächst muss man sich mit der API vertraut machen. Dann werden Startpunkte festgelegt, in diesem Fall ein Twitter-Profil, und die Einstellungen werden angepasst. Schließlich können die Daten erhoben und exportiert werden.

Schritt 1: Installation und Datenbank anlegen Installieren Sie sich die aktuelle Version von Facepager, indem Sie der für Ihr System passenden Anleitung auf GitHub folgen.[28] Dort werden Installationsdateien für Windows und Mac bereitge-

[28] Siehe Jünger und Keyling (2022; https://github.com/strohne/Facepager).

stellt. Facepager wurde mit Python entwickelt – unter Linux (und auch sonst) können Sie Facepager aus dem Quellcode laufen lassen, haben dann aber keine vorregistrierten Zugänge zu den Social-Media-Plattformen.

Öffnen Sie Facepager und legen sie über NEW DATABASE (in der Menüleiste) eine neue Datenbank an. In dieser werden alle erhobenen Daten gespeichert. Facepager sichert dabei den Stand der Datenbank fortlaufend automatisch – Sie müssen nicht extra speichern.

Schritt 2: Preset laden und Einstellungen anpassen Um einen schnellen Einstieg in die Datenerhebung zu bekommen, können Sie ein bereits angelegtes Preset über den Menüpunkt *Presets* laden. Um die Twitter-Follower eines Profils zu erheben, von dem Sie den Namen kennen, wird zunächst die ID des Profils benötigt. Dazu können Sie das Preset „Lookup user data" für die API-Version 2 laden.[29] Bevor Sie auf APPLY klicken, lesen Sie sich zunächst die Beschreibung des Presets durch. Hier sind die Funktionsweise und weitere Einstellungsmöglichkeiten beschrieben. Außerdem ist die offizielle Dokumentation des Endpunktes verlinkt. Sobald Sie das Preset anwenden, werden alle grundlegenden Einstellungen in das Query Setup (Abb. 7.8) übertragen.

Aus den Angaben im Query Setup baut Facepager selbstständig die URL für die Abfrage zusammen, indem Base path, Resource und Parameter aneinandergesetzt werden. Ein Grundprinzip von Facepager ist dabei die Arbeit mit Platzhaltern. Diese werden in spitzen Klammern < > angegeben und während der Datenabfrage durch Werte aus den Nodes ersetzt. Wenn man mehrere Nodes hinzufügt oder bereits erhoben hat, können Erhebungsschritte automatisiert werden, indem nacheinander bei gleichbleibenden Einstellungen immer andere Nodes-Daten in die Platzhalter eingefügt werden. In Abb. 7.8 werden zwei Platzhalter verwendet: `<Object ID>` wird durch den Wert `uni_greifswald` ersetzt und der Platzhalter `<username>` wird wiederum durch `<Object ID>` ersetzt.[30] Im Ergebnis wird über die gesetzten Einstellungen folgende URL zusammengebaut:

[29] Das Preset setzt die Registrierung einer App auf Twitter voraus (siehe unten). Wenn Sie keine eigene App registrieren wollen, können Sie eines der Presets für die Version 1 der API ausprobieren.

[30] Die doppelte Verwendung von Platzhaltern wäre nicht unbedingt nötig und wird hier nur aus Gründen der Übersichtlichkeit verwendet, damit die Object ID über einen Parameter angegeben werden kann.

```
https://api.twitter.com/2
/users/by/username/uni_greifswald
?user.fields=id%2Cname%2Cusername%2Cdescription
```

Die `%2C`-Zeichen kodieren dabei das Komma, diese sogenannte Prozentkodierung (engl. *percent encoding*) wird für Sonderzeichen in URLs verwendet (siehe Abschn. 2.1). Wie die einzelnen Bausteine der URL aussehen müssen und welche zusätzlichen Angaben möglich wären, sind der offiziellen Dokumentation von Twitter zu diesem Endpunkt entnommen (Abb. 7.9). Dort können Sie sehen, dass die Angaben unter „Endpoint URL" in Facepager auf den Base path und die Res-

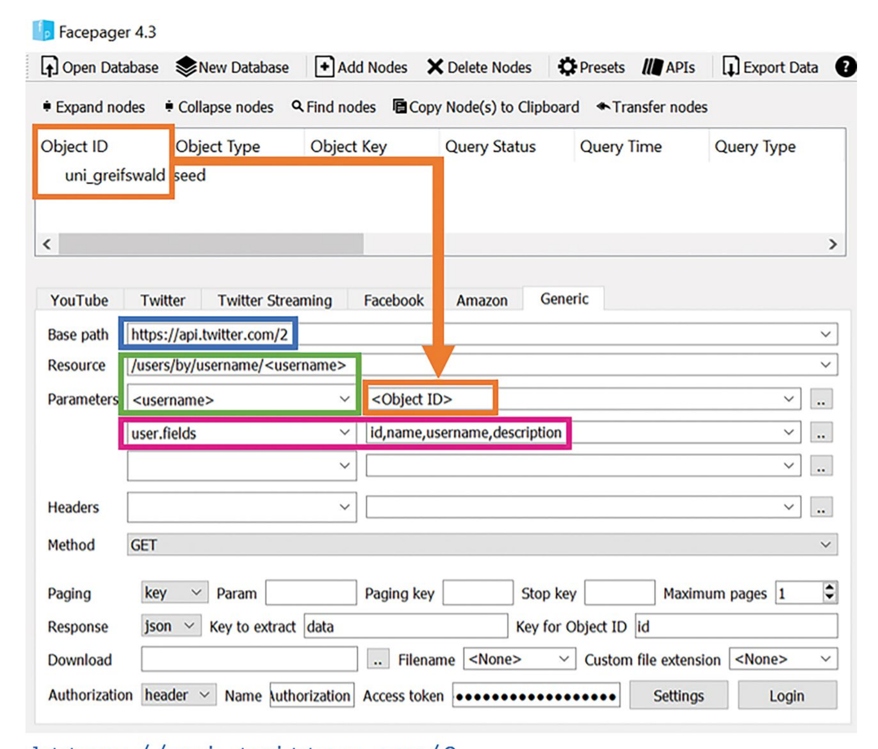


Abb. 7.8 Einstellungen zur Erhebung von Profilinformationen auf Twitter. Der Platzhalter <Object ID> wird bei der Erhebung durch den Startknoten ersetzt. (Quelle: eigene Darstellung)

GET /2/users/by/username/:username

Returns a variety of information about one or more users specified by their usernames.

Endpoint URL

`https://api.twitter.com/2/users/by/username/:username`

Authentication and rate limits

Authentication methods supported by this endpoint	OAuth 2.0 Authorization Code with PKCE OAuth 1.0a is also available for this endpoint. OAuth 2.0 App-only
Rate limit	App rate limit (Application-only): 300 requests per 15-minute window shared among all users of your app

Path parameters

Name	Type	Description
`username` REQUIRED	string	The Twitter username (handle) of the user.

Query parameters

Name	Type	Description
`tweet.fields` OPTIONAL	enum (`attachments` , `author_id` , ...)	This fields parameter enables you to select which specific Tweet fields will deliver in each returned pinned Tweet.

Example responses

Default fields Optional fields

```
1 {
2   "data": {
3     "id": "2244994945",
4     "name": "Twitter Dev",
5     "username": "TwitterDev"
6   }
7 }
```

Response fields

Name	Type	Description
`id` DEFAULT	string	Unique identifier of this user. This is returned as a string in order to avoid complications with languages and tools that cannot handle large integers.
`name` DEFAULT	string	The friendly name of this user, as shown on their profile.

Methode (GET) und Pfad des Endpunkts mit dem Platzhalter ":username"

Endpoint URL: Basis-URL für den Endpunkt

Hinweise zu **Einschränkungen** und Authentifizierung

Path Parameters: Angaben, die im Pfad der Basis-URL benötigt werden. Unter jedem Parameter steht, ob er notwendig oder optional ist.

Query Parameters: Parameter, die mit ? an die URL angehängt werden können.

Beispiel, wie das **Ergebnis** der Anfrage aussieht.

Manchmal wird auch aufgeführt, wie eine **Beispielanfrage** aussieht, das heißt eine URL mit allen Parametern.

Response Fields: Erläuterung der Daten (Key-Value-Paare), die im Ergebnis enthalten sind

Abb. 7.9 Dokumentation eines Endpunkts zur Abfrage von Profilinformationen (gekürzt). (Quelle: Twitter (2022e; https://developer.twitter.com/en/docs/twitter-api/users/lookup/api-reference/get-users-by-username-username))

source aufgeteilt werden und dass der Pfad einen mit Doppelpunkt gekennzeichneten Platzhalter für den Benutzernamen enthält. Die Angaben zum Parameter „user.fields“ sind aus den optionalen Query-Parametern in der Dokumentation zu entnehmen. Öffnen Sie die Dokumentation einmal selbst und finden Sie heraus: Wie können Sie erheben, zu welchem Zeitpunkt der Account erstellt wurde?[31] Sobald Sie solche Dokumentationen verstehen, können Sie sich selbst beliebige Anfragen zusammenbauen.

Schritt 3: Authentifizierung Um die API nutzen zu können, müssen Sie sich zunächst authentifizieren.[32] Dieser Prozess gestaltet sich je nach API und teils sogar je nach Version der API unterschiedlich. Das Preset im Beispiel nutzt die zweite Version der API, bei der Sie selbst eine App registrieren müssen. Eine App zu registrieren bedeutet dabei nicht – auch wenn es zunächst so aussieht –, dass Sie unbedingt selbst ein Programm schreiben müssen. Stattdessen kann das wörtlich verstanden werden, dass sie also eine Applikation und damit die Verwendung oder den Gebrauch bei Twitter registrieren. Dies können Sie auf der Seite für Entwickler:innen von Twitter beantragen, wobei der Anwendungsfall zunächst von Twitter geprüft wird.[33]

Nachdem Sie einen Zugang erhalten haben, tragen Sie im Query Setup unter *Settings* im Feld Client Id den erhaltenen API key und im Feld Client Secret den API secret key ein. Wenn Sie anschließend auf Login klicken, sollte automatisch ein persönliches Access Token generiert werden. Da dieses wie ein geheimes Passwort funktioniert, mit dem Sie die API abfragen, sind die einzelnen Zeichen in Facepager durch schwarze Punkte ersetzt.

Schritt 4: Startpunkte hinzufügen Um nun Daten zu erheben, fügen Sie über Add Nodes die Namen eines oder mehrerer Twitter-Profile als Startknoten (engl. *seed node*) hinzu. Social-Media-Accounts haben nicht nur einen Namen wie „Uni Greifswald“, sondern werden häufig über ein Handle identifiziert. Das Handle eines Profils wird in der URL sichtbar, wenn man das Profil im Browser betrachtet – es handelt sich in diesem Fall um den Teil, der direkt nach der Domain folgt: `https://twitter.com/uni_greifswald`. Darüber hinaus werden Profile über numerische IDs identifiziert, die für jeden Account einzeln vergeben werden

[31] Auflösung: Indem Sie für den Parameter „user.fields“ neben „id, name, username, description“ ebenfalls „created_at“ angeben.

[32] Genau genommen handelt es sich um zwei Schritte: Mit der Authentifizierung loggen Sie sich nur ein, danach wird geprüft, ob sie auch für den Zugriff autorisiert sind.

[33] Siehe Twitter (2020; https://developer.twitter.com/en/apply-for-access).

und die häufig nicht über den Browser sichtbar sind. Inwiefern für eine Abfrage das Handle oder die ID benötigt wird, ist in der Dokumentation festgelegt. Die Abfrage der Profilinformationen funktioniert mit dem Handle als Input, in den Profilinformationen ist die ID enthalten, mit der dann die Abfrage der Followees gelingt.

Schritt 5: Daten erheben Um nun die Daten zu erheben, klicken Sie zunächst den Knoten im Nodes View an (hier den Knoten „uni_greifswald"). Wenn Sie mehrere Knoten hinzugefügt haben, können Sie über SELECT ALL NODES im Query Setup auch alle Knoten gleichzeitig auswählen. Haben Sie die gewünschten Knoten ausgewählt, klicken Sie auf FETCH DATA. Im Status Log können Sie beobachten, wie nun die URLs für die Abfrage zusammengebaut und die einzelnen Daten erhoben werden.

Die neuen Daten werden den Startknoten untergeordnet. Klappen Sie die Startknoten auf, indem Sie auf den Pfeil links neben dem Knoten oder auf EXPAND ALL NODES klicken. Sobald Sie einen Knoten ausgewählt haben, können Sie im Data View alle erhobenen Daten inspizieren (Abb. 7.10).

Über das Column Setup passen Sie an, welche Spalten im Data View angezeigt und später exportiert werden (Abb. 7.10). Wenn Sie ein Preset geladen haben, sind bereits einige Spalten voreingestellt. Wollen Sie diese entfernen, klicken Sie auf CLEAR COLUMNS. Einzelne Spalten fügen Sie hinzu, indem sie ein Key-Value-Paar im Data View auswählen und auf ADD COLUMN klicken. Der Schlüssel dieses Paares

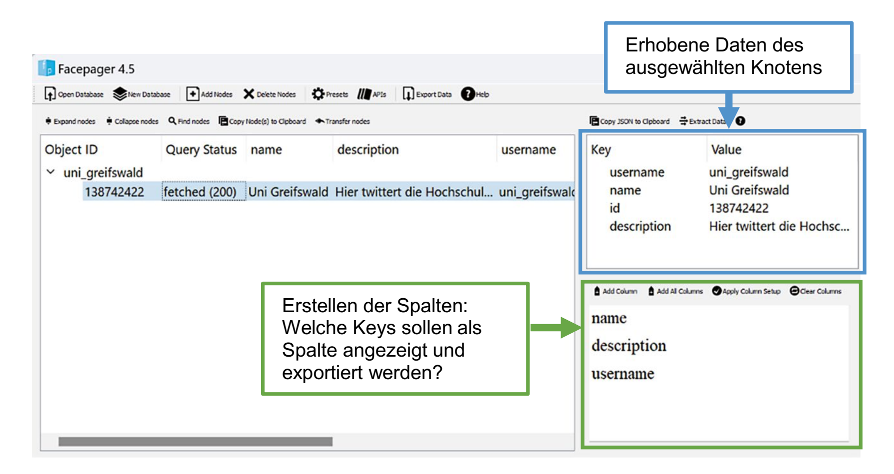


Abb. 7.10 Schritte nach der Datenerhebung mit Facepager: Daten inspizieren und Spalten anpassen. (Quelle: eigene Darstellung)

wird dadurch eine Spalte in der Tabelle im Nodes View. Wollen Sie alle Schlüssel hinzufügen, klicken Sie auf Add All Columns. Nachdem Sie die gewünschten Spalten ausgewählt und hinzugefügt haben, müssen Sie noch abschließend über Apply Column Setup die angepassten Spalten bestätigen. Scrollen Sie nun in der Nodes View nach rechts, dann sehen Sie die zusätzlichen Spalten.[34]

Wenn Sie nun die ID eines Twitter-Profils haben, können Sie die Followees dieses Profils erheben. Laden Sie dafür im nächsten Schritt das Twitter-Preset „Get followees“ (siehe Schritt 4). Bevor Sie auf Fetch Data klicken, kontrollieren Sie noch das Node level. Während sich das hinzugefügte Twitter-Handle auf der obersten Ebene (level 1) befindet, liegt die ID der vorherigen Abfrage auf der zweiten Ebene (level 2). Entweder markieren Sie für die Abfrage der Followees den Knoten mit der ID oder – insbesondere wenn Sie mehrere Startknoten eingefügt haben – Sie setzen in den Settings neben dem Query Setup das Häkchen bei Select all nodes und stellen das Node level auf 2 ein. Mit Fetch Data wird anschließend die Abfrage für alle Knoten auf der zweiten Ebene ausgeführt.

Schritt 6: Daten exportieren Überprüfen Sie nach der Erhebung, ob Sie weitere Spalten hinzufügen wollen. Um die Daten zu exportieren, wählen Sie dann die gewünschten Knoten aus und klicken auf Export Data. Wenn Sie alle Knoten exportieren wollen, können Sie das direkt im Exportfenster angeben. Dort navigieren Sie zum Arbeitsverzeichnis, legen einen Namen fest und speichern so die erhobenen Daten als CSV-Datei. CSV-Dateien enthalten tabellarische Daten, die mit allen gängigen Statistikprogrammen oder auch mit Tabellenprogrammen wie Excel, Numbers oder Calc eingelesen werden können (siehe Abschn. 3.3).

7.2.6 Automatische Bilderkennung über eine Cloud-API

Tools wie Facepager bieten eine grafische Bedienoberfläche an, sind für vielfältige APIs nutzbar und enthalten hilfreiche Funktionen etwa für umfangreichere Erhebungen und zur Parallelisierung von Anfragen. Gleichzeitig können Sie aus den Fehlern, die dabei unweigerlich auftreten, etwas über die einzelnen nötigen Schritte lernen und machen sich mit den Dokumentationen von APIs vertraut. Eine Alternative dazu stellen Packages für R oder Python dar, die auf ganz bestimmte APIs zugeschnitten sind. Sie finden solche Packages etwa für die Erhebung von Twitter-

[34] Die Platzhalter und Schlüssel können weitere Funktionen zur Transformation von Daten enthalten, was insbesondere für das Webscraping nützlich und im Facepager-Wiki erläutert ist (Jünger 2020; https://github.com/strohne/Facepager/wiki/Webscraping).

oder Telegramdaten und viele andere Social-Media-Plattformen, aber auch für die Analyse und Aufbereitung von Daten. Eine Vielzahl von Anbietern stellt APIs für die automatische Bilderkennung, die Geokodierung von Adressen oder die Sentimentanalyse von Texten zur Verfügung, die über entsprechende Packages verwendet werden können. Bilderkennung können Sie beispielsweise über die Google Cloud Vision API durchführen (Google 2022d). Auch wenn dazu im Hintergrund eine Vielzahl von Schritten nötig ist – die Authentifizierung beim Anbieter, das Hochladen der Bilder, das Formulieren der API-Anfrage, das Abrufen der Ergebnisse – lässt sich diese API beispielsweise über das R-Package *googleCloudVisionR* mit nur zwei Befehlen benutzen.

Etwas aufwendiger ist lediglich die Ersteinrichtung, denn um die API nutzen zu können, müssen Sie sich zunächst in der Google Cloud Console registrieren.[35] Die Nutzung von Clouddiensten ist in der Regel kostenpflichtig, allerdings werden häufig Freikontingente gewährt. Aktuell sind über die Google Cloud Vision API die ersten 1000 Abfragen je Monat kostenlos, behalten Sie dennoch die Kosten im Blick. Nach der Registrierung erstellen Sie in der Weboberfläche ein Projekt und aktivieren die Cloud Vision API. Im API-Bereich des Projekts erstellen Sie dann Anmeldedaten – das folgende Beispiel geht davon aus, dass Sie einen Schlüssel für ein sogenanntes Dienstkonto erstellen und auf Ihren Computer herunterladen. Die Anmeldung kann über das Package *googleAuthR* (Edmondson 2022) erfolgen, wobei Sie den Pfad zum Anmeldeschlüssel angeben:

```
library(googleAuthR)
gar_auth_service(json_file="E:/googlekey.json")
```

Prinzipiell kann das Login auch über andere Wege wie OAuth oder über Access Token erfolgen. Anschließend reicht mit dem Package *googleCloudVisionR* ein einziger Befehl, um ein Bild von dem eigenen Computer an die API zu schicken (`imagePaths`-Parameter) und daraufhin das Ergebnis der Erkennung zu erhalten:

```
library(googleCloudVisionR)

gcv_get_image_annotations(
  imagePaths = "schmetterling.jpg",
  feature = "LABEL_DETECTION",
  maxNumResults = 6
)
```

[35] Siehe Google (2022g; https://console.cloud.google.com).



mid	description	score	topicality
/m/09q2t	Brown	0.980	0.980
/m/0ql23	Pollinator	0.960	0.960
/m/0cyf8	Butterfly	0.954	0.954
/m/03vt0	Insect	0.952	0.951
/m/05s2s	Plant	0.946	0.946
/m/0zkm	Arthropod	0.944	0.944

Abb. 7.11 Das Ergebnis der Google Bilderkennung. (Quelle: Middelbos (2020))

Die API gibt nicht nur Bezeichnungen (engl. *label*) für die verschiedenen erkannten Objekte zurück, sondern auch zwei Kennwerte, mit denen die Erkennungsleistung eingeschätzt werden kann. Der Score gibt auf einer Skala von 0 bis 1 an, mit welcher Sicherheit das Objekt erkannt wurde und die Topikalität sagt aus, wie zentral das Objekt im Bild ist. Im Beispiel in der Abb. 7.11 wird mit einer recht hohen Sicherheit ein brauner Schmetterling identifiziert, der das zentrale Bildmotiv darstellt. Zusätzlich wird ein eindeutiger Bezeichner (`mid`) zurückgegeben, der für weitere Abfragen des Google Knowledge Graph verwendet werden kann. Der Google Knowledge Graph basiert auf Semantic-Web-Technologie (siehe Abschn. 3.7) und verknüpft mittels festgelegter IDs und Vokabularen eine Vielzahl an Begriffen und Wissensbeständen.

Eine API kann somit nicht nur für die Datenerhebung, sondern auch für die Vorsortierung und Aufbereitung von Daten hilfreich sein. Nach der ersten Einrichtung lassen sich auf diese Weise komplexe Analysen wie die automatische Transkription von Audioaufnahmen, die Bilderkennung oder auch die Erkennung von Hate-Speech mit beeindruckend wenig Aufwand durchführen. Die Güte der Ergebnisse hängt wesentlich von der Qualität des Ausgangsmaterials und von der Datengrundlage, mit der ein Anbieter das Erkennungsmodell trainiert hat, ab und sollte entsprechend immer überprüft werden (siehe Kap. 8). Das aber gilt grundsätzlich, wenn Sie mit automatisierten Verfahren der Datenerhebung und -analyse arbeiten.

Übungsfragen

1. Sie wollen die Likes einer Facebook-Seite über die Facebook-Graph-API mithilfe von Facepager erheben. Dafür fügen Sie „Universität Greifswald“ als neuen Knoten hinzu und laden ein entsprechendes Preset. Wieso kann diese Abfrage nicht funktionieren?

2. Finden Sie heraus: Wie viele Tweets können gleichzeitig mit einer Abfrage des Endpunktes GET statuses/user_timeline erhoben werden? Wie viele Tweets können Sie pro Tag maximal erheben?[36]
3. Was ist mit der Paginierung von Anfragen gemeint?
4. Was versteht man unter Ratenlimitierungen (engl. *rate limits*)?
5. Was ist ein Access Token und wie erhalten Sie ein Access Token?

Weiterführende Literatur

Magallanes Reyes, J. M. (2017). *Introducing data science for social and policy research. Collecting and organizing data with R and Python.* Cambridge: Cambridge University Press.

Mitchell, R. (2018). *Web scraping with Python. Collecting more data from the modern web* (2. Aufl.). Sebastopol: O'Reilly.

Munzert, S., Rubba, C., Meißner, P. & Nyhuis, D. (2015). *Automated data collection with R: A practical guide to web scraping and text mining*. Chichester: Wiley.

Nyhuis, D. (2021). Application programming interfaces and web data for social research. In: U. Engel, A. Quan-Haase, S. X. Liu & L. Lyberg (Hrsg.), *Handbook of Computational Social Science*, *Volume 2* (S. 33–45). London: Routledge.

Russell, M. A. (2014). *Mining the social web. Data mining Facebook, Twitter, Linkedin, Google+, GitHub, and more*. (2. Aufl.). Sebastopol: O'Reilly.

Literatur

Apache Software Foundation. (2021). Apache Nutch (Version 1.18) [Computer software]. https://nutch.apache.org/

Apify Technologies. (2022). Apify SDK. The scalable web crawling, scraping and automation library for JavaScript/Node.js. (Version 2.3.0) [Computer software]. https://sdk.apify.com/

Apogne. (2014). *Wait until page is loaded with Selenium WebDriver for Python,* Stack Overflow. Zugriff am 03.05.2022. https://stackoverflow.com/questions/26566799/wait-until-page-is-loaded-with-selenium-webdriver-for-python

Barbaresi, A. (2022). trafilatura: Web scraping tool for text discovery and retrieval (Version 1.2.1) [Computer software]. https://trafilatura.readthedocs.io/en/latest/

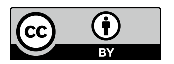
Crossref. (2022). *Crossref Unified Resource API*. Zugriff am 03.05.2022. https://api.staging.crossref.org/swagger-ui/index.html

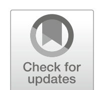
Edmondson, M., Bryan, J., de Boer, J., Richardson, N., Kulp, D. & Cheng, J. (2022). googleAuthR: Authenticate and Create Google APIs (Version 2.0.0) [Computer software]. https://cran.r-project.org/package=googleAuthR

[36] Siehe die Dokumentation des Endpunktes (Twitter 2022b; https://developer.twitter.com/en/docs/twitter-api/v1/tweets/timelines/api-reference/get-statuses-user_timeline).

Fielding, R. (2000). *Architectural styles and the design of network-based software architectures*. Dissertation. Irvine: University of California.
Fielding, R., Nottingham, M., & Reschke, J. (Internet Engineering Task Force (IETF). (Hrsg.). (2022). RFC 9110 HTTP Semantics. https://www.rfc-editor.org/rfc/rfc9110.html
Gentry, J. (2015). twitteR. R Based Twitter Client (Version 1.1.9) [Computer software]. https://cran.r-project.org/package=twitteR
Google. (2022d). *Cloud Vision API. Vision AI*. Zugriff am 30.05.2022. https://cloud.google.com/vision
Google. (2022e). *Cloud Vision-Preise*. Zugriff am 03.05.2022. https://cloud.google.com/vision/pricing
Google. (2022g). *Google Cloud Platform*. Zugriff am 03.05.2022. https://console.cloud.google.com
Haim, M. (2020). Agent-based testing. An automated approach toward artificial reactions to human behavior. *Journalism Studies*, 21(7), 895–911. https://doi.org/10.1080/1461670x.2019.1702892
Harrison, J. & Kim, J. Y. (2020). RSelenium. R Bindings for 'Selenium WebDriver' (Version 1.7.7) [Computer software]. https://cran.r-project.org/package=RSelenium
Helmond, A. (2020). *DMI Tools*. Zugriff am 05.05.2022. https://wiki.digitalmethods.net/Dmi/ToolDatabase
Jünger, J. (2020). *Facepager. Wiki*. Zugriff am 03.05.2022. https://github.com/strohne/Facepager/wiki
Jünger, J. & Keyling, T. (2022). Facepager. An application for automated data retrieval on the web. (Version 4.4.4) [Computer software]. https://github.com/strohne/Facepager/
Kearney, M. (2019). rtweet: Collecting and analyzing Twitter data. *The Journal of Open Source Software*, *4*(42), 1829. https://doi.org/10.21105/joss.01829
Middelbos, R. (2020, 24. Februar). *Schmetterling Insekt Blatt Natur*. [Bild]. Zugriff am 03.05.2022. https://pixabay.com/de/photos/schmetterling-insekt-blatt-natur-4873368/
Mozilla (2022). Firefox Browser (Version 99.0.1) [Computer software]. https://www.mozilla.org/de/firefox/new
Muthukadan, B. (2018). *Selenium with Python. WebDriver API*. Zugriff am 03.07.2022. https://selenium-python.readthedocs.io/api.html
Octopus Data. (2022). Octoparse. Easy Web Scraping for Anyone (Version 8.5.2) [Computer software]. https://www.octoparse.com/
Ooms, J. (2014). The jsonlite Package: A Practical and Consistent Mapping Between JSON Data and R Objects. http://arxiv.org/pdf/1403.2805v1
Ooms, J. & McNamara, J. (2021). writexl. Export Data Frames to Excel 'xlsx' Format (Version 1.4.0) [Computer software]. https://cran.r-project.org/package=writexl
Pal, J., Koncz, T., Varkoly, B., Lukacs, P., Kocsis, E. & Teschner, F. (2020). googleCloudVisionR. Access to the 'Google Cloud Vision' API for Image Recognition, OCR and Labeling (Version 0.2.0) [Computer software]. https://cran.r-project.org/package=googleCloudVisionR
Pandas development team. (2022a). Pandas (Version 1.4.3) [Computer software]. *Zenodo*. https://doi.org/10.5281/zenodo.3509134
Pirogov, S. (2022). Webdriver Manager for Python (Version 3.5.4) [Computer software]. https://pypi.org/project/webdriver-manager/
Postman. (2022). Postman (Version 9.16.0) [Computer software]. https://www.postman.com/
Python Software Foundation. (2022c). Python (Version 3.10.5) [Computer software]. https://docs.python.org/3/index.html
RapidMiner. (2021). RapidMiner (Version 9.10.0) [Computer software]. https://rapidminer.com/
RatSWD. (2019). *Big Data in den Sozial-, Verhaltens- und Wirtschaftswissenschaften: Datenzugang und Forschungsdatenmanagement.* Berlin: RatSWD. https://doi.org/10.17620/02671.39

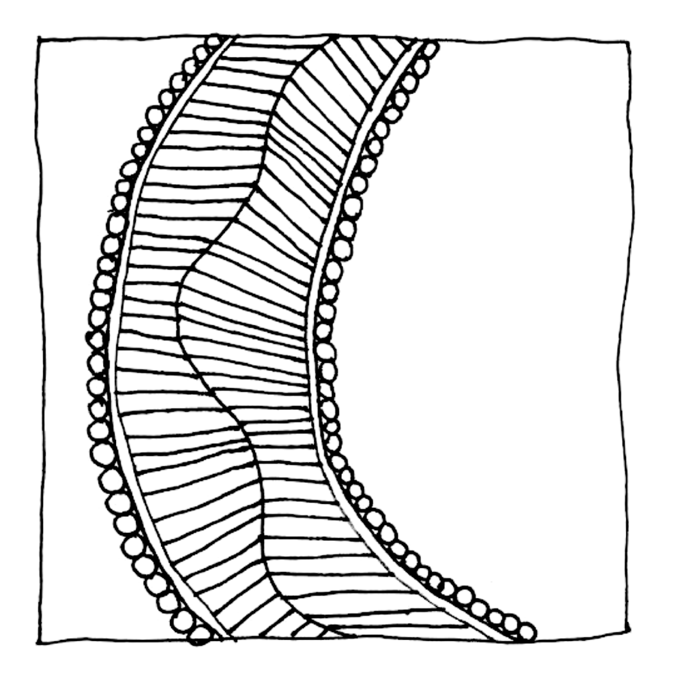
Reitz, K. (2022). *Requests: HTTP for Humans. The User Guide*. Zugriff am 03.05.2022. https://requests.readthedocs.io/
Richardson, L. (2020). *Beautiful Soup Documentation*. Zugriff am 03.05.2022. https://www.crummy.com/software/BeautifulSoup/bs4/doc/
Richardson, L. (2022). Beautiful Soup [Computer software]. https://www.crummy.com/software/BeautifulSoup
Roche, X. (2017). Httrack. Website Copier (Version 3.49-2) [Computer software]. https://www.httrack.com/
Roesslein, J. (2020). Tweepy. Twitter for Python! [Computer software]. https://github.com/tweepy/tweepy
ScrapeBox. (2022). ScrapeBox (Version 2.0) [Computer software]. http://www.scrapebox.com/
Statistical Cybermetrics Research Group. (2016). SoSciBot. Web crawler and link analyser for the social sciences and humanities (Version 4) [Computer software]: University of Wolverhampton. http://socscibot.wlv.ac.uk/
Stenberg, D. (2022). curl. A command line tool and library for transferring data with URL syntax [Computer software]. https://curl.se/
Telegram. (2020). *MTProto Mobile Protocol*. Zugriff am 28.06.2020. https://core.telegram.org/mtproto
Thelwall, M. [Mike], & Stuart, D. (2006). Web crawling ethics revisited: Cost, privacy, and denial of service. *Journal of the American Society for Information Science and Technology, 57*(13), 1771–1779. https://doi.org/10.1002/asi.20388
ThoughtWorks. (2022). Selenium [Computer software]. https://www.selenium.dev/
Twitter. (2020). *Developer platform*. Zugriff am 30.05.2022. https://developer.twitter.com/en/apply-for-access
Twitter. (2022a). *Follow, search, and get users. API reference GET followers/ids*. Zugriff am 03.05.2022. https://developer.twitter.com/en/docs/twitter-api/v1/accounts-and-users/follow-search-get-users/api-reference/get-followers-ids
Twitter. (2022b). *Get Tweet timelines. API reference: GET statuses/user_timeline*. Zugriff am 04.05.2022. https://developer.twitter.com/en/docs/twitter-api/v1/tweets/timelines/api-reference/get-statuses-user_timeline
Twitter. (2022c). *Platform overview. Get started with the Twitter Developer Platform*. Zugriff am 03.05.2022. https://developer.twitter.com/en/docs/platform-overview
Twitter. (2022e). *Users lookup. API reference GET /2/users/by/username/:username*. Zugriff am 03.05.2022. https://developer.twitter.com/en/docs/twitter-api/users/lookup/api-reference/get-users-by-username-username
Twitter. (2022f). *Follow, search, and get users. API reference GET followers/list*. Zugriff am 03.05.2022. https://developer.twitter.com/en/docs/twitter-api/v1/accounts-and-users/follow-search-get-users/api-reference/get-followers-list
Web Scraper. (2021). WebScraper. Making web data extraction easy and accessible for everyone (Version 0.6.4) [Computer software]. https://webscraper.io/
Wickham, H. (2019b). *Tidyverse. R packages for data science*. Zugriff am 24.04.2022. https://www.tidyverse.org/
Wickham, H. (2022a). *httr: Tools for Working with URLs and HTTP*. Zugriff am 05.05.2022. https://httr.r-lib.org/
Wickham, H. (2022b). rvest. Easily Harvest (Scrape) Web Pages [Computer software]. https://rvest.tidyverse.org/
Wikipedia. (2021, 8. Oktober). *Liste auflagenstärkster Zeitschriften. Deutschland*. Zugriff am 03.05.2022. https://de.wikipedia.org/wiki/Liste_auflagenst%C3%A4rkster_Zeitschriften#Deutschland
Zyte. (2022). Scrapy (Version 2.6.1) [Computer software]. https://scrapy.org/


BY



Maschinelles Lernen

8



J. Jünger, C. Gärtner, *Computational Methods für die Sozial- und Geisteswissenschaften*, https://doi.org/10.1007/978-3-658-37747-2_8

Zusammenfassung

Dieses Kapitel führt in die Grundlagen von Machine-Learning-Verfahren ein. Sie lernen, was überwachte von unüberwachten Lernverfahren unterscheidet, wie man Emotionen mit einem Künstlichen Neuronalen Netz erkennt und wie umfangreiche Textkorpora mittels Topic Modeling erschlossen werden.

Im Online-Repositorium unter https://github.com/strohne/cm finden Sie begleitend zum Kapitel weitere Materialien, auf die wir im Text mit ☛ verweisen.

Schlüsselwörter

Maschinelles Lernen · überwachte Lernverfahren · unüberwachte Lernverfahren · Klassifikation · Regression · Künstliche Neuronale Netze · Multilayer Perceptron · Bilderkennung · Scikit-learn · Confusion Matrix · Document-Term-Matrix · Topic Modeling · Latent Dirichlet Allocation · Quanteda

Wenn Datenmengen unüberschaubar groß sind oder im Laufe der Zeit viele neue Fälle auftauchen, gerät man schnell an zeitliche Grenzen – es ist nicht möglich, jeden Fall manuell anzuschauen und auszuwerten. Bei komplexen Analyseansätzen, die auf mehreren zehntausenden von Fällen basieren und eine Vielzahl von Einflussfaktoren berücksichtigen, ist es auch nicht zielführend, jeden Einflussfaktor getrennt zu betrachten. In solchen Fällen kann man einen Computer darauf trainieren, dass er selbst nach Lösungen für Analyseprobleme sucht. Dies ist das grundlegende Prinzip des sogenannten maschinellen Lernens (engl. *machine learning*). Dabei wird der Computer auf bestehenden Daten trainiert, sodass ein Modell zur Beschreibung dieser Daten entsteht. Das Modell kann dann auf neue Daten angewendet werden.[1] So lernt beispielsweise ein E-Mail-Programm, ob eine ankommende E-Mail Spam ist oder nicht. Und auch der Netflix-Algorithmus entscheidet durch das Auswerten der bisherigen Nutzung, welche Inhalte er einzelnen Nutzer:innen individuell empfiehlt. Die Besonderheit dieser Verfahren besteht

[1] Dieses Kapitel bietet einen Einstieg in die Grundzüge von Machine Learning. Wenn Sie Expert:in in dem Gebiet werden wollen, kann es sich lohnen, einen entsprechenden Online-Kurs zu belegen oder ein spezialisiertes Lehrbuch durchzuarbeiten. Ein Klassiker in diesem Bereich ist Andrew Ng's Machine Learning-Kurs auf Coursera, siehe Ng (2022; https://www.coursera.org/learn/machine-learning).

darin, dass die Klassifikationen und Empfehlungen nicht (ausschließlich) auf Regeln basieren, die etwa eine Redaktion entworfen hat, sondern dass diese aus bestehenden Daten abgeleitet werden.

Wenngleich Buzzwords wie künstliche Intelligenz (KI), Artifical Intelligence (AI), Künstliche Neuronale Netze (KNN) oder eben auch Machine Learning (ML) in den letzten Jahren in aller Munde sind und als innovative Technologien erscheinen, sind diese Methoden gar nicht so neu. Der Grundaufbau künstlicher neuronaler Netze, der bis heute vielfältig weiterentwickelt wurde, ist bereits vor vielen Jahrzehnten beschrieben worden (McCulloch und Pitts 1943; siehe auch Russell und Norvig 2012, S. 39). Und bereits in den 1950er-Jahren wurden Programme entwickelt, um Computern das Schachspielen beizubringen (Samuel 1959). In dieser Frühzeit des maschinellen Lernens wurde auch das Schlagwort „Künstliche Intelligenz“ erfunden, das bis heute mit teils mythischen Vorstellungen einhergeht (Natale und Ballatore 2020).

Auch wenn es deutliche Unterschiede zwischen Menschen und Maschinen gibt, sind die an menschliche Verhaltensweisen angelehnten Metaphern hilfreich, um automatisierte Verfahren zu begreifen. Eine typische Definition maschinellen Lernens geht auf Tom Mitchell zurück: „A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E“ (Mitchell 1997). Mitchell folgend wäre eine Spamerkennungssoftware ein Machine-Learning-Programm, das Spam-E-Mails erkennt (Task T), indem es durch Menschen vorkategorisierte Mails zur Hilfe nimmt (Experience E) und den Anteil der richtig eingeordneten E-Mails berechnet (Performance P). Das Grundprinzip des maschinellen Lernens wird demnach bereits seit einiger Zeit für die Datenanalyse eingesetzt, die eingesetzten Methoden haben sich aber über die Zeit weiterentwickelt – nicht zuletzt, da aktuelle Computer immer mehr Daten speichern und verarbeiten können.

Die einzelnen Ansätze können dabei grundlegend in zwei Arten von Lernverfahren eingeteilt werden, je nachdem, ob das Ergebnis im Vorhinein bekannt ist oder nicht (siehe auch Wrobel et al. 2003):[2]

[2] Mitunter wird davon als weiteres Verfahren das Reinforcement-Learning abgegrenzt, bei dem das Verhalten von Agenten (z. B. Autos oder Menschen) simuliert wird. Wenn sich etwa automatisierte Autos einen Weg durch unwegsames Gelände bahnen sollen, werden die Verhaltensweisen des Systems bestraft oder bestärkt, wodurch sich die zukünftige Performance verbessert.

- **Überwachtes Lernen:** Ausgehend von einem Datensatz ist bereits die korrekte Antwort für einige Fälle bekannt, zum Beispiel wenn Spam-E-Mails erkannt werden sollen. Basierend auf diesem Wissen wird dem Computer ein Algorithmus beigebracht, durch den er weitere korrekte Antworten vorhersagen kann (siehe Abschn. 8.1).
- **Unüberwachtes Lernen**: Beim unüberwachten Lernen ist vorab nicht bekannt, wie die Lösung am Ende aussehen wird. Ein Beispiel wäre, dass die gesamte Berichterstattung eines Jahres nach Themen sortiert wird, ohne dass es bereits eine Liste von Themen gibt. Stattdessen sucht der Computer selbst in den Daten nach Mustern und gruppiert die Daten anhand der gefundenen Zusammenhänge (siehe Abschn. 8.2).

Unter die Sammelbegriffe des überwachten und unüberwachten Lernens lässt sich eine Vielzahl von spezifischen statistischen Verfahren bzw. Machine-Learning-Verfahren einordnen, einige überwachte Verfahren sind in Tab. 8.1 aufgeführt, typische unüberwachte Verfahren finden sich in Tab. 8.2. In diese beiden Bereiche wird in den folgenden Kapiteln einmal anhand automatisierter Bilderkennung (Künstliches Neuronales Netzwerk) und einmal anhand automatisierter Texterkennung (Topic Modeling) eingeführt. Hat man die Grundprinzipien verstanden, lassen sich die Modellierungsansätze innerhalb eines Bereichs vergleichsweise leicht austauschen. Um etwa von der linearen Regression zu einem künstlichen neuronalen Netz zu wechseln, sind dank entsprechender R- und Python-Packages nur wenige Änderungen am Code nötig.

8.1 Überwachte Lernverfahren

Maschinelle Lernverfahren sind universell einsetzbar, sei es bei der Analyse von Texten, Bildern oder Zahlen. Dennoch ist es notwendig, sich mit den speziellen Anforderungen der inhaltlichen Fragestellungen auseinanderzusetzen. Je mehr man über die Domäne weiß, – also je besser man sich beispielsweise mit der Bibel auskennt, wenn man Bibeltexte analysieren will –, umso besser können passende und effiziente Verfahren ausgewählt werden und umso leichter sind die Ergebnisse am Ende interpretierbar. Bei überwachten Lernverfahren ist Vorwissen unabdingbar, denn die richtige Antwort auf eine Frage muss für einen Teil der Daten bereits

Tab. 8.1 Beispiele für überwachte Machine-Learning-Verfahren

Verfahren	Beschreibung
Lineare Regression	Anhand einer Regressionsgleichung wird der Output durch eine Gewichtung der Input-Werte vorhergesagt. Die lineare Regression sagt metrische Werte voraus, das heißt Kommawerte wie Preise, Lebensalter oder die durchschnittlich erwartbaren Likes zu einem Posting.
Logistische Regression	Die Regressionsgleichung wird in die Sigmoid-Funktion gekapselt, wodurch der Output einen Wert zwischen 0 und 1 annimmt, es werden Wahrscheinlichkeiten vorausgesagt. Mit logistischer Regression wird klassifiziert, beispielsweise um traurige von glücklichen Gesichtsausdrücken zu unterscheiden.
Künstliche Neuronale Netze	Künstliche Neuronale Netze erweitern den Regressionsansatz, um komplexere Modelle zu ermöglichen. Das Netz besteht aus einer Input-Schicht mit den Eingabewerten und einer Output-Schicht mit den vorhergesagten Ergebnissen. Dazwischen werden verdeckte Schichten geschaltet, die durch Gewichtung der Input-Werte den Output ermitteln. Sie eignen sich zur Vorhersage metrischer und kategorialer Daten und werden etwa zur Bild- und Texterkennung eingesetzt. Speziell für Bilder eignen sich Convolutional Neural Networks (CNN) und für die Spracherkennung werden häufig Rekurrente Neuronale Netze eingesetzt (RNN). Erstere bilden räumliche und letztere zeitliche Strukturen ab. Transformer-Modelle gehen noch einen Schritt weiter, indem sie automatisch auswählen, auf welche Ausschnitte des Datenstroms das Modell reagiert.
Naive Bayes	Naive-Bayes-Klassifikatoren verwenden bedingte Wahrscheinlichkeiten zur Klassifikation. Beispielsweise könnte ein Indikator für Spam-Mails sein, dass häufiger als in anderen E-Mails die Formulierung „click here" auftritt. Dieser Ansatz wird oft als effizientes Referenzmodell verwendet, um dann im nächsten Schritt mit komplexeren Modellen noch bessere Ergebnisse zu erzielen.
Support Vector Machine	Support-Vector-Machines werden häufig zur Klassifikation eingesetzt, wenn sich die Objekte zwar gut voneinander unterscheiden lassen, aber nicht linear separierbar sind. Anwendungsfälle wären zum Beispiel die Erkennung von Schriftzeichen in Bildern, weil sie komplexe optische Muster enthalten.
Decision Trees	Entscheidungsbäume dienen der Klassifikation, wobei die Entscheidung für eine Klasse auf eine Kombination von erlernten Regeln zurückgeführt führt.

Quelle: Eigene Darstellung

Tab. 8.2 Beispiele für unüberwachte Lernverfahren

Verfahren	Beschreibung
Clusteranalyse	Die Clusteranalyse gruppiert ähnliche Objekte, das heißt, Fälle werden zusammengefasst. So können beispielsweise verschiedene Internetnutzungstypen wie aktive und passive Nutzer:innen gebildet werden. Es gibt eine Vielzahl von Verfahren für unterschiedliche Anwendungsfälle, verbreitet sind die hierarchische Clusteranalyse (z. B. das Ward-Verfahren) und partitionierende Verfahren (z. B. k-means). Die Verfahren unterscheiden sich darin, ob die Anzahl der Cluster vor oder nach der Analyse festgelegt wird, wie die Ähnlichkeit zwischen Objekten berechnet und auf welche Art die Objekte gruppiert werden.
Faktorenanalyse	Die Faktorenanalyse reduziert ähnliche Merkmale auf latente Konstrukte, das heißt es werden typischerweise nicht Fälle, sondern Variablen zusammengefasst. Hat man beispielsweise mit einem psychologischen Fragebogen verschiedene Verhaltensweisen erfasst, lassen sich diese auf Persönlichkeitsmerkmale wie Extraversion oder Offenheit zurückführen. Auch der Begriff Faktorenanalyse umfasst im weiteren Sinne viele verschiedene Verfahren. Ein besonders einfaches Verfahren ist die Hauptkomponentenanalyse (engl. *principal component analysis*, PCA).
Topic Modeling	Nimmt man an, dass gemeinsam auftretende Wörter auch ähnliche Bedeutungen haben – „Aktivismus“ tritt etwa im Kontext von „Politik“ auf – und dass Dokumente mit ähnlichen Wörtern ähnliche Themen behandeln, dann können Texte entsprechend gruppiert werden. Beim Topic Modeling wird versucht, die Themenstruktur hinter einer Menge von Texten zu rekonstruieren, das heißt automatisch Themen zu erkennen.
Autoencoder	Autoencoder sind eine Variante Künstlicher Neuronaler Netze. Sie reduzieren wie die Faktorenanalyse einen hochdimensionalen Merkmalsraum auf wenige Merkmale, die sich dann wieder auf die Eingabedaten anwenden lassen. Auf diese Weise können etwa die wesentlichen Merkmale eines Bildes extrahiert werden (= encoder), um dann einen Malstil darauf anzuwenden und Fotos im Stil von Van Gogh zu erzeugen (= decoder). Auch das Rauschen in Bildern lässt sich mit diesen Techniken reduzieren.

Quelle: Eigene Darstellung

vorher bekannt sein. Ausgehend davon wird ein Modell trainiert, das weitere korrekte Antworten vorhersagen kann. Wenn beispielsweise Emotionen in Bildern erkannt werden sollen, muss bereits vorab für einen Teil der Bilder erfasst sein, welche Emotionen diese abbilden. Bei der Modellierung wird dann überprüft, inwiefern vorhergesagte und tatsächliche Antworten übereinstimmen – der Prozess wird also zu einem gewissen Grad durch Menschen überwacht.

Überwachte Lernverfahren lassen sich in zwei unterschiedliche Arten unterteilen – je nachdem, welche Werte vorhergesagt werden:

- Wenn der vorherzusagende Output aus **kontinuierlichen Werten** besteht, handelt es sich um ein **Regressionsproblem**. Liegen beispielsweise Merkmale von Kunstwerken vor und es soll das Schöpfungsjahr vorhergesagt werden, so kann der vorhergesagte Wert ein beliebiges Datum zwischen 40.000 vor unserer Zeitrechnung und heute sein.
- Soll der Output als **diskreter Wert** vorliegen, wird ein **Klassifikationsproblem gelöst**. Das bedeutet, die Ergebnisse werden in Schubladen gesteckt. Kunstwerke könnten in Epochen kategorisiert werden und in den Schubladen mit den Bezeichnungen (engl. *labels*) „altägyptische Malerei" oder „Renaissance" landen.

Ein sehr universelles Modell für überwachtes Lernen ist ein Künstliches Neuronales Netz (KNN; engl. *artificial neural network*, ANN). Die Idee hinter Künstlichen Neuronalen Netzen besteht darin, menschliches Denken zu imitieren, indem das Gehirn mit seinen Milliarden von verknüpften Neuronen durch mathematische Funktionen nachgebaut wird. Auch wenn es sich dabei nur um eine Metapher handelt, werden diese Verfahren als künstliche Intelligenz bezeichnet. Dadurch lassen sich Aufgaben bewältigen, die für uns Menschen auf den ersten Blick einfach erscheinen, für einen Computer aber nur durch komplizierte Funktionen lösbar sind – wie das Erkennen von Objekten und Texten in Bildern.

Formal beschrieben wurde das Modell eines künstlichen Neurons bereits 1943 von Walter Pitts und Warren McCulloch (McCulloch und Pitts 1943). Pitts und McCullochs sogenannte Threshold Logit Unit gewichtet Input-Werte und gibt bei Überschreiten eines festgelegten Schwellenwerts einen Output zurück – wie ein menschliches Neuron, das infolge eines genügend starken Reizes ein Signal weiterleitet. Praktisch implementiert wurde diese Idee erstmals in der einfachen Verarbeitungseinheit des sogenannten *Perzeptrons* (Rosenblatt 1958). Im Gegensatz zum ursprünglichen Modell eines künstlichen Neurons und auch zu einfachen statistischen Regressionsverfahren werden mittlerweile mehrere Schichten und Verfahren miteinander kombiniert, was als Deep Learning (LeCun et al. 2015) bezeichnet wird. So werden Künstliche Neuronale Netze für die Bilderkennung beispielsweise als sogenannte *Multilayer Perceptrons* oder *Convolutional Neural Networks* (LeCun und Bengio 2003) modelliert.

Ganz wesentlich für die Modellierung ist ein Algorithmus, der Backpropagation genannt wird (Rumelhart et al. 1986). Dieser Algorithmus passt die Verbindungen zwischen den Neuronen so lange an, bis eine möglichst optimale Konfiguration gefunden ist. Auch wenn Künstliche Neuronale Netzwerke dadurch sehr mächtig und prinzipiell auf fast jedes Regressions- oder Klassifikationsproblem anwendbar sind, ist diese Vorgehensweise nicht immer ratsam, da sie teilweise sehr rechenintensiv ist. Ist beispielsweise bereits bekannt, dass sich Bots oder Spam durch ganz bestimmte Merkmale auszeichnen, können einfachere Verfahren schneller und besser nachvollziehbar zu den gleichen Ergebnissen kommen. Auch die erkenntnistheoretische Zielsetzung bestimmt die Wahl des Verfahrens. Wenn das Ziel einer Analyse, wie häufig in sozial- oder geisteswissenschaftlichen Studien, darin besteht, die Welt möglichst nachvollziehbar zu beschreiben oder grundlegende Zusammenhänge zu prüfen, sind einfachere Modelle mit wenigen Parametern hilfreich. Steht dagegen weniger das Modell als vielmehr die korrekte Erkennung von Gegenständen oder Wörtern im Vordergrund, führen komplexere Verfahren wie Künstliche Neuronale Netzwerke häufig besser zum Ziel (James et al. 2013, S. 24). Es sollte aber nicht unterschätzt werden, dass für gute Erkennungsleistungen eine intensive Entwicklungsarbeit nötig ist. Für die verschiedenen Anwendungsfälle finden sich etablierte Softwarepackages wie OpenCV (Objekterkennung; Bradski 2000) und Tesseract (Texterkennung; Smith 2022), die auch für Einsteiger:innen geeignet sind.

Wie Künstliche Neuronale Netze funktionieren und modelliert werden, um damit Machine-Learning-Probleme zu lösen, wird in diesem Kapitel behandelt. Dafür soll ein einfaches Künstliches Neuronales Netz trainiert werden, dass die Emotionen ‚neutral' und ‚glücklich' in Portraitfotos erkennt.[3] Dies geschieht anhand von drei grundlegenden Schritten, die in jedem Machine-Learning-Projekt durchgeführt werden müssen:

1. **Aufbereitung:** Die Daten werden üblicherweise in eine Matrixform gebracht. Dabei wird jeder Fall, also beispielsweise jedes Bild, in einer Zeile aufgeführt und in den Spalten werden die Merkmale erfasst, sodass sich für jedes Pixel eine Spalte ergibt. Zusätzlich wird beim überwachten Lernen eine Spalte für die vorgegebenen Ergebnisse erstellt, in diesem Fall mit den Emotionskategorien (darin unterscheiden sich überwachte und unüberwachte Verfahren).

[3] Die Materialien für dieses Kapitel – die Bilder sowie nachfolgend beschriebenen Skripte – finden sich im ☛ Repositorium. Die Bilder, die in diesem Kapitel verwendet werden, stammen aus dem FER-2013-Datensatz (Goodfellow et al. 2013), der über die Plattform Kaggle heruntergeladen werden kann, siehe Sambare (2020; https://www.kaggle.com/msambare/fer2013). Der ursprüngliche Datensatz umfasst etwa 36.000 Bilder mit sieben verschiedenen Emotionen; in diesem Kapitel werden nur die Kategorien ‚neutral' und ‚glücklich' berücksichtigt.

2. **Modellierung:** Um Fälle zu clustern oder Kategorien vorherzusagen, wird ein Modell erstellt. Dieses Modell enthält dann etwa die Angabe, welches Pixel wie wichtig für die Vorhersage einer Emotion ist. Um zu diesem Modell zu kommen, wird ein Algorithmus zur Verarbeitung der vorgegebenen Daten festgelegt.
3. **Validierung:** Die Güte des Modells wird eingeschätzt, indem beispielsweise die durch das Modell vorhergesagten mit den manuell vorgegebenen Kategorien verglichen werden. Selbst wenn ein Modell die vorliegenden Daten gut beschreibt, muss abgesichert werden, dass es auch auf neue Fälle anwendbar, also generalisierbar, ist.

Die einführend vorgestellte Definition maschinellen Lernens entspricht genau dieser Einteilung: Mithilfe von Erfahrung (aufbereitete Daten) bewältigt ein Programm eine Aufgabe (Modellierung) mehr oder weniger gut (Validierung und Vorhersage). Diese Schritte sind dementsprechend nicht auf die Klassifizierung von Bildern beschränkt, sondern grundlegend für die Analyse von Daten.

8.1.1 Datenaufbereitung

Damit Autos selbst fahren, Postkarten an die richtige Adresse geliefert werden, krankhafte Anomalien in Röntgenbildern oder Gegenstände auf Fotos automatisiert erkannt werden können, muss einem Computer das „Sehen" beigebracht werden (engl. *artificial vision* oder auch *computer vision*). Menschen haben von klein auf gelernt, den visuellen Reizen in Bildern sinnvolle Bedeutungen zuzuschreiben und können demnach eine Statue von einem Wandteppich unterscheiden, den Verlauf einer Straße erkennen oder Emotionen in den Gesichtern anderer Personen deuten. Damit ein Computer ebenso Fotografien nach vorgegebenen Emotionen sortieren kann, wird er mit vorsortierten Bildern trainiert.

Dafür werden Bilder zunächst in eine für den Computer verarbeitbare Matrixform[4] gebracht, wobei jedes Bild in einer Zeile erfasst ist (Abb. 8.1). Ganz ähnlich werden auch Texte, Videos oder Tonaufnahmen für die automatisierte Datenanalyse aufbereitet, wenn jedes geschriebene Wort oder jedes Audiosample als ein einzelnes Feature begriffen wird. Zur Beschreibung der Matrizen haben sich Konventionen ausgebildet. Die Anzahl aller Bilder, anhand derer einem Computer das Sehen beigebracht werden soll, wird mit m bezeichnet; bei 12.180 Fotos gilt dem-

[4] Eine Einführung in die Datenformate Vektor und Matrix findet sich in Abschn. 3.1, einen Einblick in mögliche Transformationsverfahren von Vektoren und Matrizen gibt Abschn. 4.2.4.

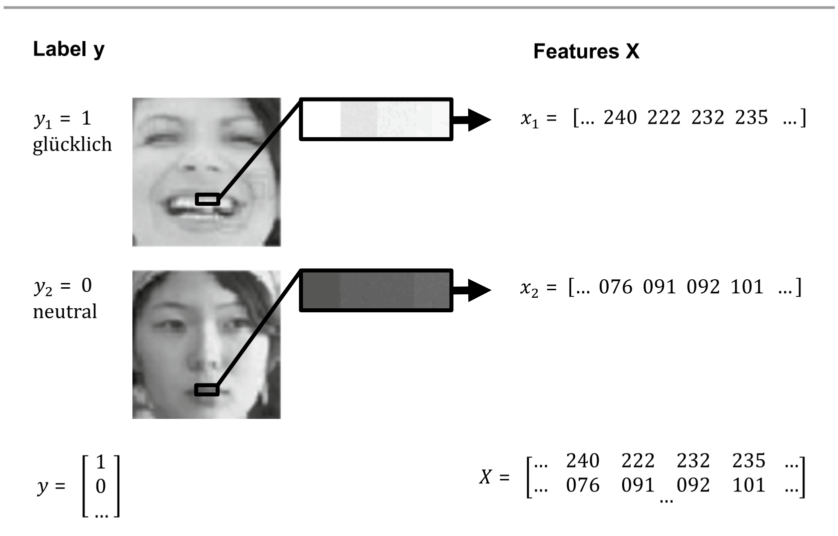


Abb. 8.1 Transformation von Bildern in die Feature-Vektoren x_i und die Label y_i. (Quelle: eigene Darstellung basierend auf FER-2013. (Goodfellow et al. 2013; Sambare 2020))

nach $m = 12.180$. Jeder einzelne Fall, also jedes Foto, wird mit all seinen Merkmalen (engl. *features*) in einen Vektor x_i transformiert, das bedeutet, das Bild i wird durch eine Liste von Pixeln dargestellt. Liegen Fotos in Graustufen vor, nimmt jedes Pixel einen Wert zwischen 0 und 255 an – durch diesen Wert wird die Helligkeit des Pixels ausgedrückt. Hat das Bild beispielsweise die Dimension 48 × 48 Pixel, kann dieses Bild durch den Vektor x_i mit insgesamt 2304 Pixel-Werten genau beschrieben werden. Die Anzahl der Werte in einem Vektor wird mit n bezeichnet, im Beispiel wären dies also $n = 2304$ Features.

Die zum Bild zugehörige Emotion, die der Computer später selbst erkennen soll, wird durch den Wert y_i erfasst. Häufig wird dabei die Referenzkategorie als $y = 0$ bezeichnet und die vorhergesagte Kategorie als $y = 1$. Die Zahlen 0 und 1 sind in diesem Fall selbst festgelegte Label für die Emotionskategorien ‚neutral' und ‚glücklich'. Wenn mehrere Kategorien gleichzeitig vorhergesagt werden sollen, beispielsweise die Gesichtsausdrücke wütend, lächelnd oder ängstlich, lassen sich diese in jeweils einer mit 0/1 kodierten Spalte erfassen, wobei dann nur die zutreffende Spalte eine 1 enthält und alle anderen Spalten eine 0. Diese Form wird als One-Hot-Kodierung bezeichnet. In der Statistik ist es zudem üblich, die Spalte der Referenzkategorie wegzulassen, da sich diese automatisch ergibt, wenn alle anderen Spalten auf 0 stehen. Dann spricht man von Dummy-Kodierung. Die Werte aller Vektoren x_i bilden zusammen die Matrix X mit der Dimension m *x* n. Ergänzend dazu beinhaltet der Vektor y bei überwachten Lernverfahren die vorhergesagten Werte.

8.1.2 Modellierung

Um nun Bilder zu klassifizieren, lernen Computer, von einzelnen Pixelwerten auf die abgelichteten Emotionen zu schließen. So könnte etwa die Helligkeit des Pixels an der Stelle, an der üblicherweise der Mund ist, darauf hindeuten, ob die Zähne zu sehen sind (helles Pixel, großer Wert) oder ob der Mund geschlossen ist (dunkles Pixel, kleiner Wert). Kann man die Zähne sehen, würde das bedeuten, dass der Mund offen ist – die abgebildete Person lacht dann wahrscheinlich oder sieht überrascht aus. Ein Computer weiß natürlich nicht, dass es sich bei dem ausgewählten Pixel um den Mund handelt. Allerdings kann er den Zusammenhang erkennen, wenn ausgerechnet bei Bildern mit dem Label ‚glücklich' wiederholt helle Pixel an der gleichen Stelle auftreten und dies als entscheidendes Feature für die Klassifikation neuer Fotos aufnehmen. Die Zusammenstellung der Features und das Verfahren, wie von den Features auf die Label geschlossen wird, stellt also eine vereinfachende Modellierung der Wirklichkeit dar.

Ein Pixel allein reicht nicht aus, um einen Gesichtsausdruck als glücklich zu klassifizieren. Um genauere Vorhersagen zu treffen, werden alle Pixel verwendet, in der Hoffnung, dass sie sich auf emotionsrelevante Bildbereiche beziehen – wie aufgerissene oder entspannte Augen, eine gerunzelte oder glatte Stirn oder auch hoch- oder zusammengezogene Augenbrauen. Um festzulegen, welchen Einfluss ein Pixel hat, also wie wichtig es für einen bestimmten Gesichtsausdruck ist, wird es mit dem Parameter Theta θ gewichtet. Ein Pixel mit einem hohen Gewicht hat einen starken Einfluss auf den gesuchten Ausdruck, ein niedriges Gewicht bedeutet, dass der Gesichtsausdruck nicht unbedingt aus dem Pixel ableitbar ist. Jedes der entsprechenden Pixel wird dabei einzeln gewichtet und schließlich werden die gewichteten Pixelwerte addiert. Die Gewichte können auch negativ sein und für mehrere Pixelwerte kombiniert werden, um beispielsweise abzubilden, dass ein offener Mund nur für Freude steht, wenn nicht gleichzeitig die Stirn gerunzelt ist. Aus der Summe der gewichteten Pixelwerte wird schließlich mithilfe der Sigmoid-Funktion[5] eine Wahrscheinlichkeit berechnet, mit der die gesuchte Emotion vorliegt. Das hat zur Folge, dass der errechnete Wert zwischen 0 und 1 liegt. Ein Wert von 0,76 bedeutet beispielsweise, dass das Foto mit einer Wahrscheinlichkeit von 76 % die gesuchte Emotion zeigt. Setzt man die bisherigen Erläuterungen zusammen, entsteht folgende Gleichung, bei der x für Pixel, θ für die Gewichte und y für die Wahrscheinlichkeit stehen:

[5] Alternativ werden andere sogenannte Aktivierungsfunktionen eingesetzt, etwa die Relu- oder die Softmax-Funktion.

$$y = sigmoid(\theta_1 \times x_1 + \theta_2 \times x_2 + \theta_3 \times x_3 + \ldots)$$

Das ist die Formel der logistischen Regression,[6] einem klassischen statistischen Ansatz zur Wahrscheinlichkeitsvorhersage, der vielen klassifizierenden Machine-Learning-Ansätzen zugrunde liegt. Dieses Modell stößt allerdings an seine Grenzen, wenn komplexere, nichtlineare Zusammenhänge zwischen den einzelnen Pixeln und der vorherzusagenden Kategorie überprüft werden sollen. Künstliche Neuronale Netze erweitern diese Idee deshalb, indem der Input nicht direkt in den Output überführt wird. Stattdessen werden sogenannte verdeckte Schichten (engl. *hidden layer*s) dazwischengeschaltet, die aus einzelnen Neuronen bestehen (Abb. 8.2). Die Neuronen der Input-Schicht entsprechen den Pixeln X und sagen die Aktivität der Neuronen in den folgenden Schichten voraus, wobei die Neuronen zwischen zwei angrenzenden Schichten im einfachsten Fall alle vollständig miteinander verknüpft sind. Wie groß die Schichten und wie sie miteinander verbunden sind, wird als Netzwerkarchitektur bezeichnet. Ein typisches Multilayer-Perzeptron enthält eine verdeckte Schicht, die weniger Neuronen als

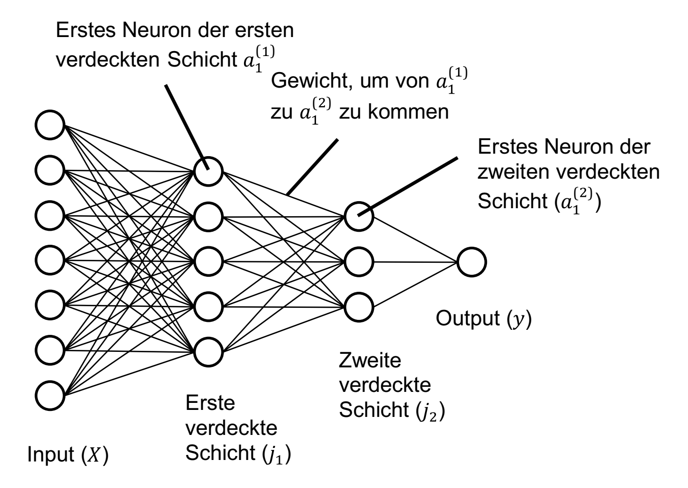


Abb. 8.2 Architektur eines neuronalen Netzes. (Quelle: eigene Darstellung)

der Input, aber mehr Neuronen als der Output enthält. Mehrere verdeckte Schich-

[6] Genau genommen wird noch ein Basisterm mit dem Gewicht θ_0 hinzugefügt, der die Basiswahrscheinlichkeit für die vorherzusagende Kategorie erfasst. Eine gut verständliche Einführung in die logistische Regression findet sich in Field et al. (2012).

ten können komplexere Zusammenhänge erfassen, sind aber gegebenenfalls auch rechenintensiver und verkomplizieren das Modell. Die letzte Schicht entspricht dem Label y. So werden die gewichteten Inputs über mehrere Regressionsgleichungen an die nächste Gleichung weitergegeben. Die Werte werden also von der Input-Schicht ausgehend immer eine Schicht weiter bis zum Output nach vorne geschoben – weswegen dieser Vorgang auch als *feed forward* bezeichnet wird.

Durch die verschiedenen Schichten und Neuronen sind kompliziertere Vorhersagen möglich, da die Erkenntnisse der einzelnen Gleichungen miteinander kombiniert werden können. So könnten beispielsweise in der ersten verdeckten Schicht einige Neuronen dafür zuständig sein, den Mund zu erkennen, während andere Neuronen die Augen entdecken. In der zweiten Schicht könnten dann die unterschiedlichen Kombinationen ermittelt werden: Ein offener Mund und entspannte Augen zeigen wahrscheinlich eine glückliche Person, die gerade lacht, während ein offener Mund und aufgerissene Augen eher die Emotion ‚überrascht' abbildet. Ein geschlossener Mund und entspannte Augen zeigen eventuell einen neutralen Gesichtsausdruck. Dieses Abdecken unterschiedlicher Kombinationen, sogenannte Interaktionseffekte, wäre durch eine einzige Regressionsgleichung, wie sie oben aufgeführt ist, nur umständlich möglich.

Das Geheimnis Künstlicher Neuronaler Netze besteht nun darin, die richtigen Gewichte für alle Neuronen zu finden, damit der Output auch korrekt vorhergesagt wird. Dazu wird das Künstliche Neuronale Netz mit den manuell kategorisierten Bildern trainiert, was als maschinelles Lernen bezeichnet wird. Die Gewichte werden zunächst (zufällig) vorbelegt und für alle Bilder werden Vorhersagen erstellt. Davon ausgehend wird der Fehlerterm (auch Kosten genannt) berechnet – also wie weit die Vorhersage mit den bestehenden Gewichten von den tatsächlichen Daten entfernt ist. Die Gewichte werden anschließend immer weiter verschoben. Wird der Fehlerterm dabei kleiner, sind die neuen Gewichte besser geeignet. Dieser Prozess wird so lange wiederholt, bis der Fehlerterm nicht mehr (bedeutend) kleiner wird, auch wenn am Ende in der Praxis immer ein Fehler übrigbleibt. Damit nicht geraten werden muss, in welche Richtung die Gewichte zu verschieben sind, wird ein Backpropagation genannter Algorithmus eingesetzt, der die Gewichte schrittweise rückwärts vom Output bis zum Input optimiert. Die genaue Umsetzung dieses Algorithmus (Rumelhart et al. 1986) muss nicht selbst implementiert werden, denn zahlreiche Bibliotheken in R und Python stellen fertige Funktionen für maschinelles Lernen bereit.

8.1.3 Validierung

Was genau in den einzelnen verdeckten Schichten passiert, ist nur schwer erfassbar, weswegen neuronale Netze auch als Blackbox angesehen werden können.[7] Das ist eine typische Eigenschaft von Machine-Learning-Ansätzen, bei denen am Ende vor allem zählt, ob das Modell funktioniert. Um einzuschätzen, wie gut das Modell seinen Zweck erfüllt und die richtigen Emotionen erkennt, muss es validiert werden. Ist das Ergebnis nicht zufriedenstellend, kann man beispielsweise Variablen ergänzen, weglassen, kombinieren, die Architektur des Modells verändern oder gar das neuronale Netz gegen einen anderen Ansatz wie eine Support Vector Machine tauschen.

Confusion Matrix Um einschätzen zu können, wie gut ein Modell funktioniert, berechnet man den Anteil der Fehlklassifikationen. Ein einfaches Maß, um die Güte eines Modells zu bestimmen, ist die sogenannte *Accuracy* – die Fehlklassifikationsrate. Als Prozentzahl wird angegeben, welcher Anteil der vorhergesagten Werte mit den tatsächlichen Werten übereinstimmt. Eine Accuracy von 79 % würde beispielsweise aussagen, dass 79 % der Bilder korrekt durch das Modell klassifiziert wurden. Die restlichen 21 % haben durch das Modell eine Emotion zugeordnet bekommen, die nicht der tatsächlichen Emotion entspricht. Die Accuracy kann allerdings irreführend sein, da sie die Klassifikationsrate besonders bei unbalancierten Daten (bzw. schiefen Verteilungen) überschätzt. Sind beispielsweise sehr wenige überraschte Bilder in einer großen Sammlung von Fotografien, kann es durchaus sein, dass das Modell alle neutralen Bilder richtig erkennt und deshalb eine hohe Accuracy aufweist. Allerdings hilft das nicht beim Erkennen der wenigen überraschten Gesichter.

Die Accuracy wird aus der *Confusion Matrix* berechnet, aus der sich weitere Gütekriterien ableiten lassen. Dazu werden in einer 2×2-Matrix die Anzahlen der tatsächlichen zu den vorhergesagten Werten ins Verhältnis gesetzt (Abb. 8.3). So wird der Anteil der richtig und falsch klassifizierten Ergebnisse bestimmt:

- **Richtig positiv:** Der Wert wurde als positiv vorhergesagt und ist auch tatsächlich positiv, zum Beispiel wurde die Emotion ‚glücklich' vorhergesagt und dabei handelt es sich auch um die tatsächlich abgebildete Emotion.

[7] Einzelne Schichten lassen sich allerdings visualisieren, um einen Eindruck zu erhalten, siehe zum Beispiel Voss et al. (2021).

		Tatsächliche Klasse	
		1 Positiv	0 Negativ
Vorhergesagte Klasse	1 Positiv	True Positives (Anzahl richtig positiver Fälle)	False Positives (Anzahl falsch positiver Fälle)
	0 Negativ	False Negatives (Anzahl falsch negativer Fälle)	True Negatives (Anzahl richtig negativer Fälle)

Abb. 8.3 Confusion Matrix. (Quelle: eigene Darstellung)

- **Falsch positiv:** Der Wert wurde positiv vorhergesagt, ist aber in Wirklichkeit negativ, beispielsweise wurde die Emotion ‚glücklich' vorhergesagt, obwohl das Bild eigentlich in die Kategorie ‚neutral' fällt.
- **Falsch negativ:** Ein Fall wird als negativ vorhergesagt, ist aber eigentlich positiv, wie wenn das Label ‚neutral' vorhergesagt wird, obwohl die Emotion eigentlich ‚glücklich' ist.
- **Richtig negativ:** Ein Fall wird korrekterweise als negativ vorhergesagt, ein als ‚neutral' klassifiziertes Bild ist auch tatsächlich ‚neutral'.

Die Accuracy berechnet sich aus dem Anteil der richtig positiven und der richtig negativen Fälle zu allen Fällen:

$$Accuracy = \frac{richtig\ positiv + richtig\ negativ}{alle\ Fälle}$$

Anhand der Confusion Matrix können darüber hinausdie Gütemaße *Precision*, *Recall* und *F1* ermittelt werden. Die *Precision* gibt an, wie viele von den als positiv vorhergesagten Werten auch tatsächlich positiv sind, also wie viele von den Bildern, die durch das Modell als ‚überrascht' klassifiziert wurden, auch zutreffend das Label überrascht haben.

$$Precision = \frac{richtig\ positiv}{richtig\ positiv + falsch\ positiv}$$

Das Maß *Recall* beantwortet dahingegen die Frage: Wie viele von den tatsächlichen positiven Werten wurden auch durch das Modell korrekt erkannt? Im Beispiel geht es um den Anteil der erkannten überraschten Bilder unter allen überraschten Bildern:

$$Recall = \frac{richtig\ positiv}{richtig\ positiv + falsch\ negativ}$$

Recall und Precision hängen zusammen. Ein hoher Recall kann leicht erreicht werden, indem einfach alle Bilder als überrascht einsortiert werden, das führt aber zu einer geringen Precision. Umgekehrt würde eine hohe Precision vorliegen, wenn nur wenige Bilder als überrascht einsortiert werden. Insofern ist bei der Modellierung darauf zu achten, worin der Anwendungsfall besteht. Eine hohe Precision sollte angestrebt werden, wenn Fehlerkennungen schädliche Folgen haben. Angenommen, Straftaten würden automatisiert ermittelt werden, dann würde eine geringe Precision viele Unschuldige treffen. Ein hoher Recall ist dagegen hilfreich, wenn sicherheitshalber Fehlerkennungen in Kauf genommen werden, etwa wenn nicht alle positiven Corona-Tests tatsächlich positiv sind, aber das Übersehen der Erkrankung zu einer weiteren Ausbreitung führen würde. Eine Kombination von Precision und Recall bietet das Maß *F1*:

$$F1 = \frac{2 \times Precision \times Recall}{Precision + Recall}$$

Anders als mit der Accuracy besteht hier nicht die Gefahr irreführender Werte, da sowohl eine geringe Precision als auch ein geringer Recall zu einem geringen F1-Wert führen. Deshalb ist dieses Maß für die Einschätzung der Modellgüte zu empfehlen.

(Kreuz)Validieren und Testen Ein entsprechend komplexes Künstliches Neuronales Netz kann die Trainingsdaten perfekt abbilden. Das ist aber nicht zielführend, denn wenn die Gewichte perfekt auf die bereits bekannten Bilder eingestellt sind, dann versagt das Netz in der Regel, wenn es unbekannte Bilder klassifizieren soll. Beim maschinellen Lernen muss deshalb beachtet werden, dass Modelle verallgemeinerbar sein sollen. Dabei können zwei Extrema auftreten, die beide vermieden werden sollten. Zum einen können Modelle überschätzt sein (engl. *overfitting*). Sie weisen dann eine sehr hohe Varianz auf. Das bedeutet, ein Modell passt perfekt auf die Daten und bildet alle Einzelfälle gut ab, ist aber völlig ungeeignet,

um weitere Daten vorherzusagen. Zum anderen können Modelle auch unterschätzen (engl. *underfitting*). Sie haben in diesem Fall einen hohen Bias, das heißt, sie sind nicht genau oder komplex genug und liegen bereits bei der Vorhersage der Trainingsdaten oft daneben.

Aus diesen Gründen müssen Modelle bereits während des Trainings validiert werden. Dazu wird der Datensatz gleich zu Beginn in drei Teile unterteilt (Abb. 8.4). Trainiert wird das Künstliche Neuronale Netz nur an einem Teil der Bilder (m_{train}). Anschließend wird das trainierte Modell an neuen Bildern, dem sogenannten Validierungsdatensatz ($m_{validierung}$), überprüft. Indem das trainierte Modell an neuen Bildern getestet wird, kann man feststellen, ob es auch andere Fotos richtig erkennt und somit verallgemeinerbar ist und nicht nur für einen ganz speziellen, nämlich den trainierten Datensatz, die richtige Vorhersage trifft. Sind in dem ursprünglichen Sample beispielsweise keine Brillenträger:innen vertreten, wird sich das Modell schwer tun, solche richtig zu klassifizieren.

Erzielt das Modell in der Validierung noch keine zufriedenstellenden Ergebnisse, muss überlegt werden, ob die Architektur verändert oder ob mehr Trainingsdaten eingespielt werden sollten (siehe unten). Anschließend wird erneut mit dem Trainingsdatensatz trainiert und am Validierungsdatensatz getestet. Dieser Prozess wird so lange wiederholt, bis man ein möglichst präzises Modell erhält.[8] Sind die optimalen Parameter gefunden, wird das Modell abschließend an einem letzten Teildatensatz, dem Testdatensatz (m_{test}), überprüft. Wichtig ist es, den Testdatensatz nur einmal zum Schluss und nicht bereits zur Validierung heranzuziehen. Ansonsten fließen bereits vorab die Eigenschaften der finalen Testdaten in das Modell ein. Ist das Modell anschließend geeignet, kann es auf neue Daten angewendet werden.

Trainingsdaten	Validierungs-daten	Testdaten

Abb. 8.4 Verhältnis von Trainings-, Validierungs- und Testdaten. (Quelle: eigene Darstellung)

[8] Dieser Suchprozess kann automatisiert werden, indem eine Liste mit Parametern festgelegt, jedes der dadurch bestimmten Modelle trainiert und dann das Modell mit der besten Performance ausgewählt wird. Diese Vorgehensweise wird *Grid Search* genannt. So kann etwa die Größe der ersten verdeckten Schicht in 100er-Schritten von 100 bis 10.000 variiert werden, um die optimale Einstellung zu finden.

<table>
<tr><td colspan="3">Trainings-daten</td><td>Validierungs-daten</td><td rowspan="4">Test-daten</td></tr>
<tr><td colspan="2">Trainings-daten</td><td>Validierungs-daten</td><td>Trainings-daten</td></tr>
<tr><td>Trainings-daten</td><td>Validierungs-daten</td><td colspan="2">Trainings-daten</td></tr>
<tr><td>Validierungs-daten</td><td colspan="3">Trainings-daten</td></tr>
</table>

Abb. 8.5 Aufteilung eines Datensatzes für eine Kreuzvalidierung. (Quelle: eigene Darstellung)

Wenn zum Beispiel in R oder Python für die Text- oder Bildanalyse Packages verwendet werden, die mit vortrainierten Modellen arbeiten, dann sollte man sich immer auch die dazu veröffentlichten Gütemaße anschauen – bei der Bewertung in sozial- oder geisteswissenschaftlichen Kontexten kann man sich in etwa an den in der Statistik für Reliabilitätstests etablierten Bereichen orientieren, dabei gelten Klassifikationen mit Reliabilitätswerten ab ungefähr 0,8 als zuverlässig (Neuendorf 2017, S. 168). Es kommt aber auf den konkreten Anwendungsfall an, wo man die Grenze ansetzt. Wenn es um Menschenleben geht, ist eine Rate von 20 % Fehlklassifikationen sicher nicht akzeptabel.

Für das Verhältnis der einzelnen Teildatensätze werden häufig die Anteile 60 % Trainings-, 20 % Validierungs- und 20 % Testdaten gewählt. Dabei entsteht jedoch das Dilemma, dass man möglichst viele Daten für die einzelnen Schritte haben möchte: Je mehr Daten in das Trainieren des Modells einfließen, desto präziser kann das Modell werden. Gleichermaßen gilt jedoch auch: Je mehr Daten für die Validierung zur Verfügung stehen, desto zuverlässiger ist diese. Besonders problematisch ist, wenn der Datensatz insgesamt nur wenige Fälle aufweist. Eine Lösung für dieses Problem bietet die *Kreuzvalidierung*. Dabei wird der Trainingsdatensatz gemischt und in k Teile unterteilt, beispielsweise in vier Teile.[9] Daraus ergeben sich vier Kombinationen von Trainings- und Validierungsdatensätzen (Abb. 8.5). Auf jeder Kombination wird anschließend das Modell berechnet und eine Validierung durchgeführt. Die Gesamtgüte errechnet sich aus der durchschnittlichen Leistung in allen Durchgängen. Die Kreuzvalidierung nimmt durch die multiple Berechnung mehr Zeit in Anspruch, bietet dafür aber genauere und sicherere Ergebnisse.

[9] Diese Art der Kreuzvalidierung wird auch k-fache Kreuzvalidierung (engl. *k-fold cross-validation*) genannt.

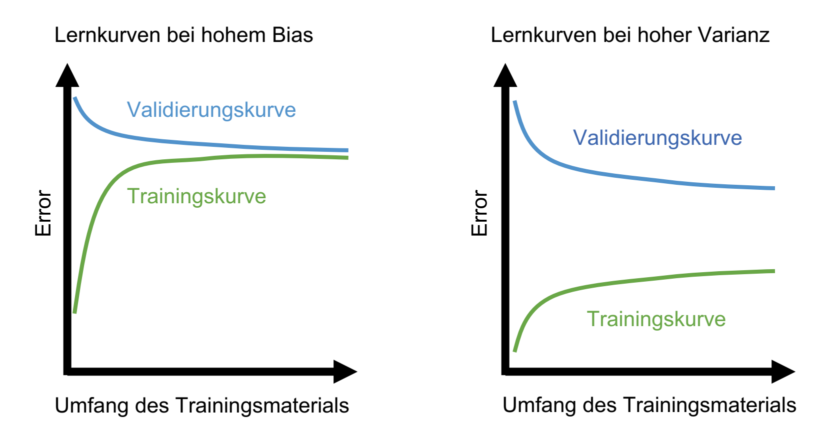


Abb. 8.6 Lernkurven bei Bias (underfitting) und Varianz (overfitting). Bei der Abbildung handelt es sich um idealisierte Lernkurven. Dargestellt ist in diesem Fall der Fehlerterm des Lernalgorithmus. Links ein hoher Fehler sowohl beim Trainieren als auch beim Validieren. Rechts ein hoher Fehler beim Validieren. (Quelle: eigene Darstellung)

Lernkurven Was ist zu tun, wenn ein Modell nicht gut funktioniert? Aufschluss darüber können Lernkurven geben, in denen einzelne Gütemaße wie der F1-Wert oder auch der beim Trainieren berechnete Fehlerterm im Zeitverlauf abgebildet werden (Abb. 8.6). Auf der x-Achse wird der Trainingsfortschritt abgebildet, mit jedem Feed-Forward- bzw. Backpropagation-Durchgang erweitert sich das verwendete Trainingsmaterial.[10] Auf der y-Achse wird die Performance des Modells gemessen. Entweder wird der Fehlerterm des Lernalgorithmus betrachtet – also wie stark die Vorhersagen des Modells von den tatsächlichen Werten abweichen – oder man visualisiert einzelne Gütemaße wie die Accuracy oder den F1-Wert. Indem man dann die Kurven je für Trainings- und Validierungsdaten erstellt, können Verbesserungsmöglichkeiten abgeschätzt werden.

Um die richtigen Schlüsse aus den Lernkurven zu ziehen, muss man sich zunächst vor Augen halten, was das dargestellte Maß aussagt und wie eine optimale Kurve aussehen sollte. Wird beispielsweise der Fehlerterm für jeden einzelnen

[10] Für das Erstellen von Lernkurven werden verschiedene Varianten verwendet, die ggf. unterschiedlich interpretiert werden müssen. So können a) die einzelnen Durchgänge (engl. *iteration*) beim Optimieren des Modells oder b) Epochen (engl. *epoch*), nach denen alle Trainingsdaten jeweils einmal durchlaufen wurden, dargestellt werden. Mitunter werden auch c) vollständige Modelle für unterschiedliche Größen von Trainingsdatensätzen berechnet, deren Größen auf der x-Achse angegeben werden.

Durchgang des Trainingsalgorithmus geplottet, beginnt die ideale Lernkurve für den Trainingsdatensatz zunächst niedrig, da am Anfang wenige Fälle vorhanden sind und das Modell diese gut abbilden kann. Je mehr Fälle hinzukommen, desto größer werden auch die Fehler. Gegenläufig verhält sich die Lernkurve auf dem Validierungsdatensatz, das Modell verallgemeinert am Anfang schlecht. Im Optimalfall ist der Fehler sowohl bei Trainings- als auch Validierungsdaten gegen Ende des Trainings gering und die beiden Kurven treffen sich, das heißt, die Trainingsleistung entspricht in etwa der Validierungsleistung.

Damit lässt sich zunächst einschätzen, inwiefern das Trainingsmaterial ausreicht. Konvergieren die Kurven nicht – sie werden nicht deutlich flacher–, dann verbessert sich die Leistung mit jedem weiteren Fall immer noch weiter. In diesem Fall lohnt es sich, weitere Daten einzuspielen und den Algorithmus länger laufen zu lassen.

Liegt dagegen eine hohe Varianz (= overfitting) vor, bleibt die Lernkurve des Trainingsdatensatzes niedrig und die Lernkurve des Validierungsdatensatzes signalisiert einen hohen Fehler – das heißt, das Modell passt gut auf die trainierten Daten, lässt sich aber nicht gut auf neuen Daten übertragen. Es bleibt also eine große Lücke zwischen den beiden Lernkurven. Bei hoher Varianz sind mögliche Optionen: Die Anzahl der Variablen zu reduzieren, da weniger Variablen zu allgemeineren Modellen führen, oder mehr Trainingsbeispiele zu sammeln, durch die die Gewichte der einzelnen Variablen besser ermittelt werden können.

Ein hoher Bias (= underfitting) wird daran sichtbar, dass der Fehler insgesamt hoch bleibt – das Modell passt also weder auf die Trainingsdaten noch auf die Validierungsdaten wirklich gut. Bei hohem Bias kann es helfen, das Modell präziser zu trainieren, indem zusätzliche Variablen ausgewählt oder komplexere Modelle mit mehr Schichten eingesetzt werden.

8.1.4 Weiterführende Konzepte des maschinellen Lernens

Wenn Sie den Text bis hierhin nachvollzogen haben, dann verfügen Sie bereits über ein solides Wissen über Machine Learning – und können im folgenden Kapitel mit der praktischen Umsetzung starten. Wenn Sie tiefer in die Materie einsteigen oder eigene Anpassungen an dem im folgenden Kapitel trainierten Netz vornehmen wollen, werden Ihnen früher oder später weitere Begriffe und Konzepte über den Weg laufen:

- **Lokales und globales Minimum**: Wenn ein Modell trainiert wird, soll der Fehlerterm, also die Abweichung des Modells von den Daten, möglichst klein werden. Wird der Fehlerterm beim Trainieren nicht mehr kleiner, erreicht er ein

Minimum – ein Zeichen, dass das Modell fertig trainiert ist. Dabei kann es jedoch passieren, dass sich der Fehlerterm nur auf einem lokalen Minimum befindet, das Modell also nur einen Teil der Daten optimal beschreibt. Würde man noch etwas weitersuchen und sich dabei auch kurzzeitig auf eine Erhöhung des Fehlers einlassen, könnte möglicherweise eine noch bessere Kombination der Gewichte gefunden werden. Der niedrigste Fehlerterm, der gesucht wird, befindet sich im globalen Minimum.

- **Regularisierung:** Komplexe Netze mit vielen Gewichten passen sich gut an die Daten an und führen zu großer Varianz (= overfitting). Dieses Problem kann dadurch eingefangen werden, dass die Stärke der Gewichte θ durch einen Regularisierungsparameter Lambda λ begrenzt wird – die Gewichte werden im Grunde selbst noch einmal gewichtet. Auch Dropout ist eine mögliche Regularisierungsvariante, dabei wird für jeden Trainingsdurchgang ein Anteil der Neuronen ausgeschaltet, sodass sie sich nicht weiter anpassen können. Beide Varianten führen dazu, dass bei Klassifikationen die Entscheidungsgrenze (engl. *decision boundary*) geglättet wird.
- **Decision boundary**: Stellt man sich ein Koordinatensystem vor, in dem die Eigenschaften der Fälle eingetragen sind, dann lassen sich bei Klassifikationsproblemen die Grenzwerte, ab denen eine Kategorie zutrifft, als Kurve beschreiben. Diese Kurve wird als Entscheidungsgrenze (engl. *decision boundary*) bezeichnet.
- **Splines und Polynome**: Im Fall von Regressionsproblemen werden kontinuierliche Werte vorhergesagt, etwa die Zeit des Medienkonsums. In einem Koordinatensystem wird diese abhängige Variable auf der y-Achse eingetragen. Trägt man auf der x-Achse den Bildungsstand ein, so lässt sich der Zusammenhang (= das Modell) als Kurve bzw. bei einer linearen Regression als Gerade visualisieren, mit der die Daten optimal beschrieben werden sollen. Komplexere Kurven werden als Polynome oder Splines bezeichnet.
- **Alpha**: Alpha bestimmt die Lernrate und wird für den Trainingsalgorithmus angegeben, der die optimalen Gewichte für eine Funktion finden soll. Ein großes alpha legt fest, dass die Gewichte in großen Schritten verschoben werden sollen, ein kleines alpha verschiebt die Gewichte immer nur wenig. Große Lernraten bergen die Gefahr, dass das Optimum verpasst wird, kleine Lernraten benötigen dagegen viel Rechenzeit.
- **Feature Extraction und Feature Scaling:** Die eingesetzten Variablen können vor dem Training ggf. noch aufbereitet werden. Feature Extraction meint in der Regel, dass mehrere Variablen zusammengefasst werden (z. B. Höhe, Breite und Länge einer Statue zum Faktor Größe), dazu werden etwa Faktorenanalysen eingesetzt. Mit Feature Scaling werden die Werte in einen vergleichbaren

Bereich gebracht, damit Berechnungen schneller laufen und einzelne Variablen nicht aufgrund des Wertebereichs einen übermäßigen Einfluss auf die Modellierung nehmen.
- **Grid Search:** Die Grid Search ist dabei behilflich, die optimale Kombination von Hyperparametern – damit sind die Voreinstellungen eines Modells, etwa die Anzahl verdeckter Schichten oder die Lernrate alpha gemeint – zu finden. Dafür wird zunächst eine Liste möglicher Parameter definiert. Für jede Kombination, die sich aus der Liste ergibt, wird anschließend ein Modell trainiert. Am Ende kann man die Modelle vergleichen und das beste Modell auswählen.
- **Ensemble Learning:** Beim Ensemble Learning werden verschiedene Lernalgorithmen bzw. Modelle kombiniert, damit das Training zu einem besseren Ergebnis führt.
- **Batch:** Das Modell muss nicht auf allen Trainingsdaten auf einmal trainiert werden. Stattdessen können die Daten in sogenannte Batches, also Pakete, unterteilt werden. Hat ein Batch beispielsweise eine Größe von 200, so werden für den ersten Trainingsschritt 200 Fälle herangezogen. Für die nächste Anpassung werden neue 200 Fälle verwendet. Dadurch werden die Daten während des Trainierens gemischt, sodass sich das Modell nicht so schnell auf einzelne Merkmale einstellen kann. Die Wahrscheinlichkeit eines Overfittings wird somit kleiner.

Die Suche nach dem optimalen Modell kann eine große Herausforderung sein – die Erkennung von Emotionen allein aufgrund von Gesichtsausdrücken ist selbst für Menschen nicht trivial. Die Mühe lohnt sich aber, denn dabei lernen Sie viel über die Technologien, die mittlerweile in den Alltag Einzug gehalten haben.

8.1.5 Ein Künstliches Neuronales Netz in Python trainieren

Ein Künstliches Neuronales Netz lässt sich in wenigen Schritten in Python (siehe Abschn. 5.2) trainieren. Dies ermöglichen spezielle Programmbibliotheken, die bereits vordefinierte Funktionen für zahlreiche Machine-Learning-Szenarien bereitstellen. Dazu zählen unter anderem Scikit-learn (Pedregosa et al. 2011), Keras (Chollet 2020) und Tensorflow (Google 2022h), die auch von Einsteiger:innen einfach bedient werden können und Algorithmen für die einzelnen Schritte im Data-Mining-Prozess (siehe Kap. 4) bereitstellen. Das Modell, das in diesem Kapitel für die Klassifizierung eingesetzt wird, ist ein Künstliches Neuronales Netz der Form eines Multilayer Perceptrons. Legen Sie als erstes ein Python-Notebook an,

legen Sie dann die Daten in das Arbeitsverzeichnis (☛ *Repositorium*) und geben Sie schließlich die nachfolgenden Befehle Schritt für Schritt ein!

Schritt 1a: Bibliotheken laden Für viele Machine-Learning-Szenarien haben bereits andere Entwickler:innen Funktionen in Python geschrieben. Damit Sie auf diese Funktionen zugreifen können, müssen Sie die entsprechenden Bibliotheken laden. Die folgenden Bibliotheken werden nachfolgend für die unterschiedlichen Aufbereitungs- und Analyseschritte verwendet:

- *numpy* und *pandas* zur Verarbeitung der Datensätze (Harris et al. 2020; Pandas development team 2022a),
- *matplotlib* und *seaborn*, um Grafiken zu visualisieren (Hunter et al. 2022; Waskom 2021),
- *Pillow* (PIL) für die Verarbeitung von Bildern (Clark und Contributors 2022),
- *scikit-learn* (sklearn) für Verfahren des maschinellen Lernens (Pedregosa et al. 2011) und
- *os* für den Zugriff auf Ordner und Dateien (Python Software Foundation 2022e).

Eingebunden werden Bibliotheken über den Befehl `import`.[11] Um nicht die gesamte Bibliothek laden zu müssen, werden unten mithilfe von `from` … `import` aus den Modulen `neural_network`, `metrics` und `model_selection` nur bestimmte Klassen geladen. Durch das Importieren mit `import as` wird die Bibliothek unter einem von Ihnen festgelegten Kürzel eingelesen. Um auf eine Funktion zuzugreifen, müssen Sie dadurch nicht immer den vollständigen Namen der Bibliothek ausschreiben, sondern können einfach das Kürzel verwenden:

```
import os

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

from PIL import Image
```

[11] Wenn Sie erstmalig die aufgeführten Bibliotheken in Python verwenden, müssen Sie diese vor dem Importieren zunächst installieren. Dafür führen Sie in der Konsole den Befehl `pip install` in Kombination mit dem Bibliotheksnamen aus. Zur Installation und der Arbeit mit Bibliotheken siehe Abschn. 5.2.

```
from sklearn.neural_network import \
                            MLPClassifier
from sklearn.metrics import confusion_matrix, \
                            classification_report
from sklearn.model_selection import \
                             train_test_split
```

Schritt 1b: Bilder einlesen und aufbereiten Bevor Sie mit dem Einlesen der Bilder starten, verschaffen Sie sich einen Eindruck von den Daten. Öffnen Sie dafür den Ordner *fer2013* mit den Bildern. In den Unterordnern *happy* und *neutral* sind die Bilder abgelegt, die nachfolgend für die Emotionserkennung verwendet werden. Nachdem Sie sich einige Fotos angesehen haben, können Sie loslegen, indem Sie die Bilder einlesen und in die Matrix X und den Vektor y umformen. Dabei bietet es sich an, eine eigene Funktion zu definieren. Innerhalb der Funktion wird jedes Bild zunächst mit `Image.open()` aus den Unterordnern eingelesen, anschließend werden die Pixelwerte des Bildes mit `list()` in eine Liste umgeformt und die Pixelliste dann durch `append()` zu einer Liste mit allen bereits eingelesenen Bildern hinzugefügt:

```
def load_images(folder):
    images = []
for filename in os.listdir(folder):
    fullname = os.path.join(folder, filename)
    img = Image.open(fullname)
    img = list(img.getdata())
    images.append(img)

return images
```

Als Input-Parameter nimmt die Funktion einen Ordner entgegen. Werden die Ordner vorher in einer Liste abgelegt, können sie in einer For-in-Schleife abgearbeitet werden. Die eingelesenen Bilderlisten werden dann mit `extend()` im Objekt `X` zusammengeführt . Das Objekt `y` wird ebenfalls in jedem Durchgang der Schleife erweitert, indem für jedes Bild der zugehörige Ordnername abgelegt wird. Nach dem Einlesen werden `X` und `y` in *numpy*-Arrays konvertiert, um die Listen für die folgenden Schritte in eine Matrixform zu bringen. Geben Sie die Objekte über den Befehl `display(X)` aus, um das Ergebnis besser zu verstehen:

```
X = []
y = []

workingdir = os.getcwd() + '/fer2013/train/'
```

```
folders = ['happy', 'neutral']

for folder in folders:
    images = load_images(workingdir + folder + '/')

    X.extend(images)
    y.extend([folder] * len(images))

X = np.asarray(X)
y = np.asarray(y)
```

Schritt 1c: Trainings- und Validierungssets erstellen Bevor das Modell trainiert wird, legen Sie bereits vorab einen Teil der Daten für die spätere Validierung beiseite. Dazu teilen Sie die Daten mithilfe der Funktion `train_test_split()` auf. Diese nimmt als Input `X` und `y` entgegen und unterteilt sie in die Anteile, die Sie durch den Parameter `test_size` festlegen. In diesem Fall werden 20 % der Daten für das Validieren des Modells beiseitegelegt. Der zusätzlich angegebene Parameter `stratify` stellt sicher, dass sowohl im Trainings- als auch im Validierungsdatensatz ein ähnlicher Anteil verschiedener Emotionen enthalten ist. Über das Setzen des `random_state` stellen Sie außerdem sicher, dass die Funktion auch bei mehrmaligem Ausführen die gleiche Aufteilung vornimmt. Die Funktion gibt vier Objekte zurück: die Trainingsdaten der Bilder `X_train`, die Validierungsdaten der Bilder `X_val`, die Trainingslabel `y_train`, die Validierungslabel `y_val`.[12]

```
X_train, X_val, y_train, y_val = \
    train_test_split(
        X, y, test_size=0.2,
        stratify=y, random_state=180
    )
```

Wenn Sie das finale Modell für reale Anwendungszwecke verwenden wollen, müssten Sie zusätzlich einige Bilder als Testdatensatz (nicht zu verwechseln mit dem Validierungsdatensatz) beiseitelegen – um das Verfahren zu erlernen, ist das zunächst noch nicht nötig.

Schritt 2: Trainieren Um ein Künstliches Neuronales Netz in der Form eines Multilayer Perceptrons zu trainieren, können Sie die Klasse `MLPClassifier()`

[12] Wenn Sie überprüfen wollen, wie groß die Trainings- und Validierungsdatensätze sind, können Sie die Funktion `len()` verwenden, beispielsweise `len(X_train)`.

verwenden. Bei der Instantiierung (= aus der Klasse wird ein konkretes Objekt erstellt, siehe Abschn. 5.2) geben Sie alle Parameter mit, mit denen Sie das neuronale Netz trainieren wollen. So nimmt beispielsweise der Parameter `hidden_layer_sizes` die Anzahl der verdeckten Schichten und Neuronen entgegen. Eine Angabe von `(1000, 100)` bestimmt, dass es zwei verdeckte Schichten gibt – die erste mit 1000 und die zweite mit 100 Neuronen. Eine Angabe von `(1000, 500, 50)` würde ein Netz mit drei verdeckten Schichten festlegen.[13]

```
mlp = MLPClassifier(
    hidden_layer_sizes=(1000, 100),
    batch_size=200,
    max_iter=1000,
    n_iter_no_change=10,
    verbose=True,
    random_state=180
)
```

Der so erzeugte Klassifizierer `mlp` verfügt über eine Methode `fit()`, mit der das Training angestoßen wird. Diese nimmt als Input die Trainingsbilder und -label entgegen:[14]

```
mlp.fit(X_train, y_train)
```

Dank des Parameters `verbose=True` wird während des Trainings ausgegeben, wie sich der Fehler (engl. *loss*) – also die Differenz zwischen den Daten und dem Modell – schrittweise verschiebt. Die Werte werden gleichzeitig in der Eigenschaft `loss_curve_` abgelegt. Um eine erste Einschätzung zu erhalten, wie gut das Training funktioniert hat, erstellen Sie daraus mithilfe der Funktion `plot()` aus der Bibliothek *matplotlib* eine Lernkurve:[15]

[13] Übernehmen Sie gerne zunächst die festgelegten Parameter aus dem Codeschnipsel und trainieren Sie das Modell damit. Wenn Sie die einzelnen Angaben genauer verstehen und das Modell selbst anpassen wollen, schlagen Sie die Erläuterungen in der Hilfe nach (Scikit-learn Developers 2022; https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html).

[14] Das Trainieren von aufwendigen, vielschichtigen Modellen kann einige Zeit zum Berechnen in Anspruch nehmen. Beobachten Sie im Output, wie sich der Fehlerterm bei jedem Durchgang (engl. *iteration*) verändert.

[15] Dem Training des Künstlichen Neuronalen Netzes liegen zufallsbasierte Prozesse zugrunde. Deswegen kann es sein, dass Ihr Output anders aussieht.

```
plt.plot(mlp.loss_curve_)
plt.xlabel("epoch")
plt.ylabel("loss")
plt.show()
```

Im Optimalfall sollte die in Abb. 8.7 dargestellte Kurve immer weiter abfallen und gegen null gehen. Ein Fehler von null bedeutet, dass es keine Abweichung zwischen den Daten und den vom Modell vorhergesagten Werten gibt – das Modell bildet die Daten also ideal ab. Der Peak, der zwischen der 20ten und 40ten Epoche liegt, kann darauf hindeuten, dass der Algorithmus zuerst in ein lokales Minimum gelaufen ist, dann von neuen Trainingsbildern überrascht wurde, anschließend aber weiter konvergiert.

Schritt 3a: Validierung und Optimierung Nach dem Trainieren des Modells können Sie überprüfen, wie gut sich das Modell eignet, um von den Bildern auf die abgebildeten Emotionen zu schließen. Dafür stellt der Klassifizierer die Methode `predict()` bereit, der Sie die Trainingsdaten übergeben. Mit der Funktion `confusion_matrix()` können Sie die vorhergesagten den tatsächlichen Werten gegenüberstellen, die Umwandlung in einen Dataframe verbessert die Dar-

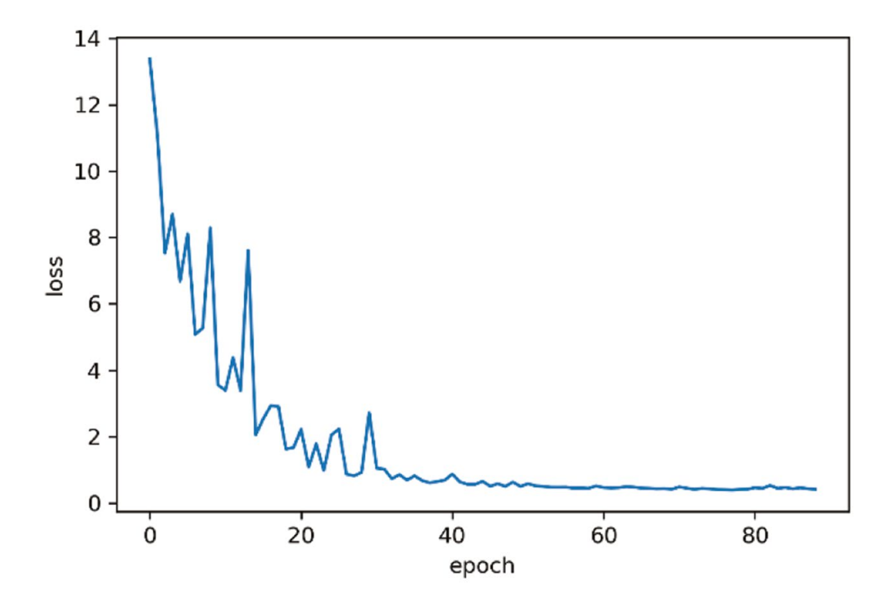


Abb. 8.7 Loss-Curve. (Quelle: eigene Darstellung)

stellung. Über die Funktion `classification_report()` können Sie daraus Gütekriterien wie die Precision, den Recall und den F1-Wert berechnen:

```
y_pred = mlp.predict(X_train)

conf = confusion_matrix(y_train, y_pred)
print(pd.DataFrame(conf))

report = classification_report(y_train, y_pred)
print(report)
```

In der Confusion-Matrix (Abb. 8.8) werden die tatsächlichen Werte in den Zeilen aufgeführt, wobei 0 in diesem Beispiel die Referenzkategorie ‚happy' und 1 die Klassifikationskategorie ‚neutral' bezeichnen.[16] Insgesamt liegen also 5772 Bilder für ‚happy' vor, was sich an den Metriken in der Spalte `support` ablesen lässt. Davon wurden immerhin 5145 richtig erkannt, was einen Recall von 89 % ergibt. Unter den als ‚happy' erkannten Bildern (erste Spalte der Confusion-Matrix) befinden sich allerdings auch 1004 in Wirklichkeit neutrale Bilder, das ergibt eine Precision von 84 %. Der daraus berechnete F1-Wert von 0,86 zeigt an, dass die Kategorie ‚happy' schon einigermaßen zuverlässig vorhergesagt werden kann. Für die Kategorie ‚neutral' hat das noch nicht ganz so gut funktioniert: Hier wurden 75 % der neutralen Fotos er-

```
Confusion-Matrix

      0     1
0  5145   627
1  1004  2968

Classification-Report

               precision    recall  f1-score   support

       happy       0.84      0.89      0.86      5772
     neutral       0.83      0.75      0.78      3972

    accuracy                           0.83      9744
   macro avg       0.83      0.82      0.82      9744

weighted avg       0.83      0.83      0.83      9744
```

Abb. 8.8 Confusion-Matrix und Gütekriterien (Python-Output). (Quelle: eigene Darstellung)

[16] Diese Zuordnung ergibt sich aus der alphabetischen Reihenfolge der Label.

kannt, wobei die als neutral klassifizierten Fotos mit 83 % recht zuverlässig auch wirklich neutral sind. Eine Accuracy von 78 % bedeutet, dass das Modell insgesamt auf den Trainingsdaten eine brauchbare Erkennungsleistung aufweist.[17]

Ob das Modell auch für weitere Anwendungsfälle geeignet ist, muss mit dem Validierungsdatensatz überprüft werden. Tauschen Sie `X_train` durch `X_val` im Aufrufvon `mlp.predict()` aus und vergleichen Sie den Output! Bewegen sich die Gütemaße im ähnlichen Bereich, lassen sich mit dem Modell auch neue Daten korrekt vorhersagen. Liegt eine hohe Diskrepanz vor, hat das Modell sich zwar gut an die Varianz der Trainingsdaten angepasst, ist im Hinblick auf weitere Daten aber überschätzt. Eine noch robustere Einschätzung der Güte erhalten Sie mittels Kreuzvalidierung auf unterschiedlich großen Teildatensätzen und der Visualisierung daraus abgeleiteter Lernkurven (☛ *Repositorium*). In der Regel führen die ersten Versuche nur zu mäßigem Erfolg und es folgt eine Phase, in der man die Parameter des Modells oder die Trainingsdaten überarbeitet.

Schritt 3b: Anwendung auf eigene Bilder Abschließend sollte das Modell an einem weiteren Testdatensatz, der nicht in das Training und in die Optimierung der Parameter eingeflossen ist überprüft werden. Daran zeigt sich, ob es auch auf neue Bilder anwendbar ist. Sind Sie zufrieden mit der Leistung des Modells, können Sie es abspeichern und ggf. auch weitergeben.[18] Das vortrainierte Modell kann nun verwendet werden, um neue Fotos zu klassifizieren und die Ergebnisse auszuwerten. Machen Sie ein Foto von sich selbst und füttern Sie das Modell damit – der Ausschnitt sollte ähnlich wie die Bilder des Datensatzes sein, auch die Größe und die Farben müssen angepasst werden. Es muss dafür gesorgt werden, alle Ausgangsdaten in eine einheitliche Form zu bringen. Die *PIL*-Bibliothek unterstützt dabei, Bilder zu verkleinern und in Graustufen umzuwandeln. Sie können folgende Zeilen in die Funktion zum Einlesen der Bilder einbauen:

```
img = img.resize((48, 48))
img = img.convert(mode='L')
```

[17] Der verwendete Datensatz war vor einigen Jahren Grundlage eines Wettbewerbes. Das Leaderboard zeigt die besten Ergebnisse (Kaggle 2013; https://www.kaggle.com/c/challenges-in-representation-learning-facial-expression-recognition-challenge/leaderboard). Der beste Teilnehmer erreichte eine Accuracy von 71 % – wobei zu beachten ist, dass er diesen Wert für alle und nicht nur für zwei Kategorien erreichte. In sozial- und geisteswissenschaftlichen Anwendungsfeldern erreichen einfache Machine-Learning-Modelle aktuell häufig eine ähnliche Performance im Bereich 70–80 %.

[18] Die Bibliothek *joblib* stellt zum Beispiel die Funktionen `dump()` zum Abspeichern und `load()` zum Einlesen bereit (Varoquaux und Joblib developers 2021).

Dann legen Sie die Bilder in den Ordner *eigenes_foto*, lesen diese mit der schon bekannten Funktion ein und lassen sich überraschen, ob Ihre Künstliche Intelligenz Sie durchschaut:

```
workingdir = os.getcwd()
X_new = load_images(workingdir + '/eigenes_foto/')
X_new = np.asarray(X_new)

y_new = mlp.predict(X_new)
print(y_new)
```

8.1.6 Weiterführende Hinweise

Das Beispiel hat Sie durch die Grundlagen überwachten maschinellen Lernens geführt. Die Vorgehensweise bleibt in der Regel gleich, unabhängig davon, ob es um die Klassifikation von Texten, die Erkennung von Objekten auf Bildern oder um Sprachverarbeitung geht. In der Praxis werden auf den verschiedenen Verarbeitungsstufen noch eine Vielzahl weiterer Entscheidungen anstehen. Insbesondere die Suche nach den optimalen Parametern für einen ausgewählten Modellierungsansatz kann einige Zeit in Anspruch nehmen. Anstatt dass Sie alle denkbaren Kombinationen von Parametern selbst austesten, führen Sie besser eine Grid Search durch. Dazu definieren Sie Listen der möglichen Parameter, zum Beispiel für zwei verdeckte Schichten alle Kombinationen der Werte zwischen 10 und 210 in 50er-Schritten:

```
from itertools import product
layers = [x for x in
            product(
              range(10, 211, 50),
              range(10, 211, 50)
            )
         ]
```

Lassen Sie sich über `print(layers)` die erstellte Liste ausgeben, um das Ergebnis besser zu verstehen. Diese Konfiguration können Sie anschließend mit `GridSearchCV` systematisch durchprüfen lassen, was allerdings aufgrund der vielen Kombinationen einige Zeit dauert:

```
from sklearn.model_selection import GridSearchCV

params = [{'hidden_layer_sizes' : layers}]
```

```
gridsearch = GridSearchCV(
    mlp,
    params,
    scoring = 'f1_micro',
    n_jobs = 6,
    verbose = 10
)

gridsearch.fit(X_train, y_train)
```

Die hinsichtlich des F1-Wertes optimale Kombination findet sich anschließend in der Eigenschaft `gridsearch.best_params`.

Ein Multilayer Perceptron eignet sich gut, um die grundlegende Vorgehensweise beim maschinellen Lernen nachzuvollziehen. Geht es um praktische Umsetzungen, kommen stattdessen aber weiter ausgearbeitete Modelle zum Einsatz, die theoretische Erkenntnisse zur jeweiligen Datenstruktur berücksichtigen. Für die Bilderkennung werden etwa Convolutional Neural Networks (CNNs) eingesetzt, die über zwei spezialisierte Arten verdeckter Schichten die räumliche Struktur von Bildern einbeziehen (LeCun und Bengio 2003):

- **Convolution**: Eine sogenannte Faltungsmatrix, oder auch Filterkernel bzw. Convolutional Window, arbeitet zunächst markante Strukturen der Bilder heraus. Die Faltungsmatrix ist deutlich kleiner als das eigentliche Foto – beispielsweise ein Gitter aus 3×3 Werten – und wird Schritt für Schritt über jeden Ausschnitt des Bildes gelegt. Dabei werden die Pixelwerte des Fotos elementweise mit den Werten des Gitters multipliziert. So entsteht ein neues Bild, in jedes Pixel sind dabei Pixelwerte der unmittelbaren Umgebung eingeflossen.
- **Pooling**: Das Ergebnis der Faltungen wird anschließend in einer Pooling-Schicht verarbeitet. Eine verbreitete Form des Poolings ist das Max-Pooling, das aus jedem 2×2 Ausschnitt der neuen Werte den größten Wert ermittelt. Dies hat zur Folge, dass die Informationen verdichtet und nur relevante Eigenschaften beibehalten werden.

Diese Vorgehensweise kommt dem menschlichen Sehen näher, denn die Pixel werden nicht wie oben als Vektor, sondern tatsächlich als zweidimensionale Anordnung behandelt. Auch für die Sprachverarbeitung bieten sich speziellere Modelle wie Rekurrente Neuronale Netzwerke (RNN) bzw. Long-Short-Term-Memory-Modelle (LSTM) an, die die sequentielle (zeitliche bzw. textliche) Anordnung von Daten berücksichtigen und gewissermaßen über ein Gedächtnis

für vorangegangene Wörter oder Laute verfügen. Sogenannte Transformer-Modelle gehen noch einen Schritt weiter, indem sie Mechanismen zur Aufmerksamkeitslenkung auf bestimmte Teile des Datenstroms einsetzen. Viele dieser Varianten können ebenfalls mit Python trainiert werden, etwa mit dem Framework Tensorflow.[19] Einen Überblick über die Performance verschiedener Modelle für den im Beispiel verwendeten Datensatz findet sich bei Khaireddin und Chen (2021).

Übungsfragen

1. Was ist der Unterschied zwischen überwachten und unüberwachten Lernverfahren?
2. Warum wird ein Datensatz in Trainings-, Validierungs- und Testdaten unterteilt?
3. Wofür stehen beim maschinellen Lernen üblicherweise die Buchstaben `X` und `y`?
4. Sie haben die Confusion Matrix für ein Modell ermittelt und erhalten folgende Werte:

```
             Tatsächlich
             |   0 |   1 |
          -----------------
Vorher-   0  |  70 |  20 |
gesagt    1  |  10 | 120 |
```

   Ermitteln Sie die Precision und den Recall. Wie schätzen Sie das Modell ein?
5. Sie haben ein Modell trainiert und erkennen nach der Validierung, dass dieses eine hohe Varianz aufweist. Was bedeutet das und was sind Ihre nächsten Schritte?
6. Was ist eine Grid Search?

8.2 Unüberwachte Lernverfahren

Nicht immer weiß man im Voraus, nach welchen Merkmalen Texte, Bilder oder andere Objekte idealerweise sortiert werden können. Häufig handelt es sich dabei um große Datenbestände, die nicht manuell gesichtet und kategorisiert werden können, beispielsweise die Berichterstattung eines ganzen Jahres, das Gesamtwerk

[19] Für einen Praxiseinstieg in CNNs mit Tensorflow und Keras siehe beispielsweise (Google 2022i; https://www.tensorflow.org/tutorials/images/cnn).

eines Autors oder einer Autorin oder die Nutzungsdaten einer umfangreichen Webseite.

Erst durch die Betrachtung aller Merkmale und Strukturen im gesamten Datensatz erschließen sich charakteristische Eigenschaften, mit denen sich einzelne Gruppen bilden lassen. Dies ist im Kern die Funktionsweise von unüberwachten Lernverfahren. So sind – anders als bei überwachten Lernverfahren (siehe Abschn. 8.1) – nicht bereits vorab für einen Teil der Daten die Kategorien bekannt. Stattdessen wird ein Lernalgorithmus angewendet, der sich selbst auf die Suche nach Strukturen und Mustern in den Daten begibt, um diese zu gruppieren. Die identifizierten Eigenschaftsbündel und Gruppen können dann nachträglich beschrieben werden, um daraus Erkenntnisse zu gewinnen oder neue Fälle einzusortieren. Auch das Bilderkennungsproblem aus dem vorherigen Kapitel könnte in einem unüberwachten Lernverfahren gelöst werden. Wenn die Label „neutral" und „happy" nicht vorab für einen Teildatensatz bekannt wären, könnte man versuchen, die Bilder nach ähnlichen Gesichtsausdrücken zusammenzufassen, um den gefundenen Gruppen anschließend nach einer manuellen Sichtung passende Label zuzuordnen. Dabei können durchaus überraschende Gruppen entstehen, die man vorab nicht erwartet hat.

Mittels unüberwachter Lernverfahren werden Daten verdichtet, sodass sich die Dimensionen eines Datensatzes reduzieren. Diese Dimensionen kann man sich insbesondere als Zeilen oder Spalten einer Matrix vorstellen:

- Um **Fälle** zusammenzufassen, etwa wenn man Nutzer:innen zu Typen gruppiert, wendet man Clusterverfahren an. Liegen die Daten als Matrix vor, werden die Zeilen zusammengefasst. Clusterverfahren lassen sich danach unterscheiden, ob die Anzahl der Gruppen vorab festgelegt werden muss oder ob sich die Anzahl aus der Analyse ergibt. Gängige Verfahren der Clusteranalyse sind das k-means-Verfahren und die hierarchische Clusteranalyse.
- Statistische Verfahren, mit denen **Variablen** zusammengefasst werden, sind die Faktorenanalyse oder die Hauptkomponentenanalyse. So können etwa mehrere untereinander korrelierende Nutzungsindikatoren wie die Häufigkeit des Logins oder die Anzahl verfasster Kommentare auf einer Webseite zu einem Faktor „Aktive Nutzung" verrechnet werden. In der Welt des Machine Learning werden die Variablen auch Features genannt und das Bestimmen eines neuen zusammengefassten Features heißt Feature Extraction. In Bezug auf Matrizen entspricht dies einem Zusammenfassen der Spalten.

Ob man mit der Verdichtung der Fälle beginnt und schließlich die Merkmale der resultierenden Gruppen beschreibt (Clusteranalyse) oder aber eine Vielzahl an Merkmalen zusammenfasst, um schließlich Fälle damit zu beschreiben (Faktoren-

analyse), hängt von der Fragestellung bzw. Analyseperspektive ab. Komplexere Verfahren verbinden diese beiden Perspektiven und modellieren bzw. simulieren den datengenerierenden Prozess (siehe Kap. 11). Dazu gehört beispielsweise das im Folgenden besprochene Topic Modeling. Im Ergebnis werden gleichzeitig die Wörter eines Textes (= Variablen) zu Themen verdichtet und es wird eine Zuordnung der Themen zu Dokumenten (= Fälle) vorgenommen.

Die Entwicklung eines unüberwachten Machine-Learning-Modells verläuft im Wesentlichen analog zu den Schritten für überwachte Lernverfahren wie sie in Abschn. 8.1 detailliert beschrieben sind. Auch hier werden die Daten erstens aufbereitet, zweitens modelliert und drittens validiert.

8.2.1 Topic Modeling

Ein weitverbreitetes Verfahren aus dem Feld unüberwachten Lernens ist Topic Modeling. Das Ziel dieses Verfahrens besteht darin, ein Textkorpus nach Themen zu strukturieren. Topic Modeling lässt sich auf vielfältige Dokumentarten wie Bundestagsreden, Social-Media-Kommentare oder Abstracts wissenschaftlicher Aufsätze anwenden.[20] Die Textmengen sollten allerdings möglichst umfangreich sein und das Vokabular in den Dokumenten sollte sich überschneiden, damit die Themen aus der Kookkurrenz von Wörtern erschlossen werden können.

Topic Modeling gibt es in verschiedenen Spielarten, das Grundprinzip der Anwendung auf Texte ist unter der Bezeichnung Latent Dirichlet Allocation (LDA) von Blei, Ng und Jordan (2003) publiziert worden. Es handelt sich um ein generatives Modell, das bedeutet dahinter liegen theoretische Annahmen, wie Texte entstehen. Stellen Sie sich vor, Sie schreiben einen Essay. Gemäß des Modells entscheiden Sie zunächst, welche Themen in welchem Umfang in Ihrem Essay angesprochen werden sollen, zum Beispiel die Bundestagswahl und der Klimawandel. Sodann beginnen Sie zu schreiben und manifestieren diese Themen, indem Sie passende Wörter verwenden. Der LDA-Algorithmus dreht diese Entstehungslogik um und versucht für die manifesten Wörter in gegebenen Dokumenten eine optimale Zuordnung[21] zu den latenten Themen zu finden: „documents are represented as random mixtures over latent topics, where each topic is characterized by a

[20] Was unter Dokumenten, Wörtern und Topics zu verstehen ist, bleibt der Interpretation überlassen. In der Biologie werden mit den gleichen Mixed-Model-Verfahren Genanalysen durchgeführt, siehe Pritchard et al. (2000), und prinzipiell lassen sich auch Musikstücke oder Bilder mittels generativer Modelle klassifizieren (Hu 2009).

[21] Daher auch der Name *Latent Dirichlet Allocation*: Die Zuordnung (engl. *allocation*) geschieht anhand der sogenannten Dirichlet-Verteilung (Blei 2012, S. 78) – eine Wahrscheinlichkeitsverteilung für multivariate Analysen.

distribution over words" (Blei et al. 2003, S. 996). Die optimale Themenstruktur ist diejenige, welche am wahrscheinlichsten die vorliegenden Dokumente hervorbringt. Dabei gibt es Grundannahmen über die Zuordnung von Themen zu Dokumenten und von Wörtern zu Themen:

- Ein Dokument zeichnet sich jeweils durch eine begrenzte Anzahl unterschiedlicher Themen aus. Ein Dokument kann also zum Großteil aus dem Thema Bundestagswahl und zu einem geringeren Anteil aus dem Thema Klimawandel bestehen.
- Ein Thema wird jeweils durch eine begrenzte Anzahl zusammenhängender Wörter repräsentiert. Das Thema Bundestagswahl könnte sich besonders durch die Wörter „Stimme" und „Wahllokal" auszeichnen, während diese Wörter beim Thema Klimawandel seltener auftreten.

Es handelt sich bei Topic Models um Mixed-Membership-Models – sowohl ein Wort als auch ein Dokument können mehreren Themen zugeschlagen werden. Die Themenzugehörigkeit wird über Wahrscheinlichkeiten angegeben – im Gegensatz zu sogenannten Hard-Clustering-Models, bei denen Daten genau einer Kategorie zugeordnet werden.

Die genaue Anzahl der Themen je Dokument und Wörter je Thema ist beim Topic Modeling nicht vorgegeben, sondern wird durch Dirichlet-Verteilungen modelliert. Die sogenannten Hyperparameter des Verfahrens geben für die verwendeten Dirichlet-Verteilungen vor, wie divers die Dokumente (alpha-Parameter α) und Themen (eta-Parameter η) sein dürfen. Dabei wird der Zusammenhang zwischen den Wörtern berücksichtigt. Ein Computer kann den Wörtern nicht selbst Bedeutungen zuschreiben. Er weiß also nicht, dass Wörter wie „Kanzlerin", „Stimme" und „Wahllokal" zu einem Thema wie der Bundestagswahl gehören könnten. Der Computer löst diese Zuordnung stattdessen über einen Algorithmus, der gleichzeitig die Wörter auf die Themen und die Themen auf die Dokumente so verteilt, dass sich eine plausible Zuordnung ergibt. Tauchen die Wörter „Kanzlerin", „Stimme" und „Wahllokal" häufig gemeinsam in verschiedenen Dokumenten auf, wird ein Zusammenhang unterstellt.

Im Prinzip müssten alle theoretisch möglichen Themenstrukturen durchgetestet werden, um die wahrscheinlichste Zuordnung zu finden. Allerdings würde diese Herangehensweise zwangsläufig an Grenzen stoßen, die Anzahl möglicher Kombinationen ist zu umfangreich, als dass sie in vertretbarer Zeit vollständig durchgerechnet werde könnte. Deshalb kommt ein zufallsbasiertes Samplingverfahren[22]

[22] Klassischerweise etwa das sogenannte Gibbs-Sampling, ein Spezialfall von Monte-Carlo-Verfahren (siehe Kap. 11 sowie Griffiths und Steyvers 2004).

zum Zuge, bei dem bestimmte Kombinationen geprüft und Schritt für Schritt verbessert werden. Zunächst legt man fest, wie viele Themen insgesamt im Korpus zu erwarten sind, diese Anzahl wird als *k* bezeichnet. Dann wird jedem Wort in jedem Text zufällig ein Thema zugeordnet. Anschließend werden die Zuordnungen solange umgeschichtet, bis sowohl die Themen auf die Dokumente als auch die Wörter auf die Themen möglichst gut den vorgegebenen Verteilungsannahmen entsprechen.

Im Ergebnis werden zwei Wahrscheinlichkeiten geschätzt. Die Wahrscheinlichkeit, mit der ein Wort zu einem Thema gehört (*beta*), ergibt sich aus der Häufigkeit, mit der ein Wort diesem Thema zugeordnet wurde.[23] Die Wahrscheinlichkeit, mit der ein Dokument zu einem Thema gehört (*gamma*), ergibt sich aus der Häufigkeit themenspezifischer Wörter im Dokument (Abb. 8.9). Auf Basis dieser Wahrschein-

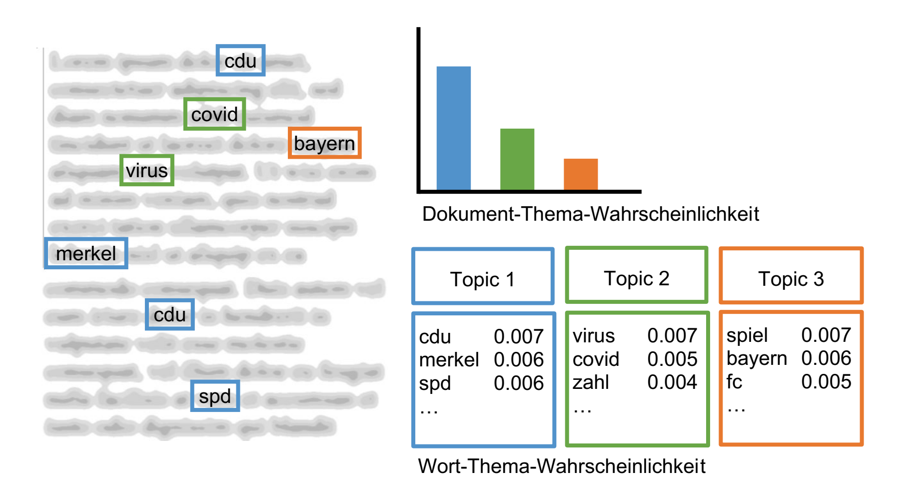


Abb. 8.9 Struktur von LDA-Topic-Models. Jedem Wort (token) wird innerhalb eines Dokuments ein Topic zugeordnet (blau, grün, orange). Daraus berechnet sich sowohl, wie stark ein Thema innerhalb eines Dokuments auftritt (rechts oben), als auch über alle Dokumente hinweg, wie häufig ein Wort (type) innerhalb eines Themas vorkommt (rechts unten). (Quelle: eigene Darstellung, nach Blei 2012, S. 78)

[23] Zu beachten ist hier der Unterschied zwischen Type und Token (siehe Kap. 9). Jedem Token (= konkretes Auftreten eines Wortes in einem Dokument) wird ein Thema zugeordnet, sodass dem gleichen Type (= das Wort unabhängig vom Auftreten) über alle Dokumente hinweg unterschiedliche Themen zugeordnet werden können.

lichkeiten kann man die typischen Wörter je Thema und die dominierenden Themen je Dokument ablesen und interpretieren. Da es sich um ein zufallsbasiertes Verfahren handelt, das zudem davon abhängt, wie die Texte aufbereitet werden, entstehen je nach Parameterwahl unterschiedliche Modelle. Dementsprechend kommt der Validierung (siehe unten) eine besondere Bedeutung zu, wenn man zuverlässig interpretierbare Ergebnisse erhalten will.

8.2.2 Topic Modeling in R

Topic-Modeling ist eine mittlerweile so weitverbreitete Technik, dass sie durch eine Vielzahl an Packages unterstützt wird. Das folgende Beispiel verwendet das R-Package *topicmodels* (Grün und Hornik 2011) für die Modellierung und Funktionen aus dem *quanteda*-Package (Benoit et al. 2018) für die Datenaufbereitung – starten Sie RStudio und installieren Sie diese sowie die anderen im jeweiligen Schritt mit dem `library()`-Befehl verwendeten Packages (siehe Abschn. 5.1).

Schritt 1: Datenaufbereitung Bevor Texte analysiert werden können, müssen sie passend zur verwendeten Topic-Modeling-Funktion aufbereitet werden. Liegen die Texte als einzelne Dateien oder in einer Tabelle vor, so werden in der Regel zunächst folgende Schritte ausgeführt (siehe Abschn. 9.1; ausführlicher siehe Maier et al. 2018):

- **Tokenisierung:** Die Texte werden in ihre einzelnen Wörter zerlegt.
- **Stemming oder Lemmatisierung:** Die einzelnen Wörter werden auf ihre Grundform zurückgeführt, sodass zum Beispiel „Wahl“ und „Wahlen“ gleichbehandelt werden.
- **Relative Pruning:** Besonders seltene und besonders häufige Wörter werden entfernt. Diejenigen Wörter werden behalten, die zur Unterscheidung der Dokumente beitragen können, also nicht überall vorkommen, gleichzeitig aber häufig genug auftreten, um typische Zusammenhänge zwischen den Wörtern sichtbar machen zu können.
- **Document-Term-Matrix (DTM):** Es wird eine Matrix erstellt, in der jedes Dokument in einer Zeile, jedes Wort in einer Spalte und die Häufigkeit des Wortes im jeweiligen Dokument in den Zellen festgehalten wird. Als alternativer Name dieser Datenstruktur findet sich häufig auch die Bezeichnung Document-Feature-Matrix (DFM).

Im folgenden Beispiel wird ein vorbereiteter Datensatz mit einem Auszug aus der deutschen Berichterstattung von Spiegel und der Bild-Zeitung verwendet, in dem die Aufbereitungsschritte bereits ausgeführt wurden (☛ *Repositorium*). Der Datensatz kann wie folgt eingelesen werden:[24]

```
library(quanteda)
dfm <- read_rds(
  "usenews.mediacloud.wm.2020.german.rds"
)
```

Liegen die Daten dagegen noch nicht in aufbereiteter Form, sondern zum Beispiel als einzelne Textdokumente im Unterordner *korpus* vor, dann lesen Sie das Korpus mit der `readtext()`-Funktion ein:

```
library(readtext)
texte <- readtext("korpus", encoding = "UTF-8")
```

Mit dem *quanteda*-Package kann die Aufbereitung mit vergleichsweise wenig Aufwand durchgeführt werden:[25]

```
# In quanteda-Corpus umwandeln
texte <- corpus(texte)

# Tokenisierung
token <- tokens(texte)

# Stemming
token <- tokens_wordstem(token)
```

[24] Der gesamte Datensatz umfasst die Berichterstattung von Nachrichtenseiten, die laut Reuters Digital News Report im Jahr 2020 am meisten genutzt wurden (Puschmann und Haim 2021; https://osf.io/uzca3/).

[25] Statt Stemming empfiehlt sich für die bessere Lesbarkeit des Ergebnisses eine Lemmatisierung, dafür können Sie beispielsweise spaCy (Honnibal und Montani 2020) verwenden, siehe Abschn. 9.3.2. Quanteda stellt zudem eigene Funktionen für das Topic Modeling bereit, siehe Quanteda (2022; https://tutorials.quanteda.io/machine-learning/topicmodel/).

```
# Document-Term-Matrix erstellen
dfm <- dfm(token)

# Relative pruning
dfm <- dfm_trim(
  dfm,
  max_docfreq = 0.1,
  docfreq_type = "prop",
  min_termfreq = 0.8,
  termfreq_type = "quantile"
)
```

Das Ergebnis ist eine hochdimensionale, aber spärlich besetzte Matrix, das heißt, die Texte werden durch eine Vielzahl an Spalten beschrieben, aber in einem einzelnen Text kommen nur wenige der Features vor (Tab. 8.3). Bei der Umwandlung gehen syntaktische und semantische Strukturen der Texte verloren, es zählt nur noch, ob oder wie oft ein Wort in einem Dokument vorkommt. Da alle Wörter zusammengeworfen werden, wird dies auch als *Bag-of-Words*-Ansatz bezeichnet. Im nächsten Schritt wird diese Dimensionalität auf eine handhabbare Anzahl an Topics reduziert. Das betrifft sowohl die Zeilen als auch die Spalten: Für jedes Wort und für jedes Dokument wird eine Zuordnung zu einem Topic vorgenommen, wobei die Zuordnung über Wahrscheinlichkeiten ausgedrückt wird.

Tab. 8.3 Auszug aus einer Document-Term-Matrix

doc_id	**china**	**twitter**	**angela**	**corona**	**trainer**	**kinder**	**…**
1192227147	0	1	0	0	0	0	
1210712025	0	0	0	0	0	0	
1380028004	0	0	0	0	0	0	
1380028010	0	0	0	0	0	0	
1380027997	0	0	0	0	0	0	
1380027982	0	0	0	0	0	0	
1380027978	0	0	1	0	0	0	
1380027970	0	0	0	0	0	0	
1380027962	0	1	0	0	0	0	
…							

Basis: rund 35.000 Dokumente (Zeilen) und 90.000 Wörter (Spalten). (Quelle: eigene Darstellung)

Schritt 2: Modellierung Nach der Aufbereitung wird das LDA-Topic-Model mit der Funktion `LDA()` trainiert. Die Funktion nimmt als erstes die zuvor erstellte Document-Term-Matrix entgegen. Zudem müssen Sie über den Parameter `k` festlegen, zu wie vielen Topics die Wörter und Dokumente gruppiert werden sollen. Ob der gewählte Wert sinnvoll ist, werden Sie erst später durch eine Validierung des Ergebnisses feststellen können. Unten finden Sie zudem Hinweise dazu, wie Sie die Entscheidung datenbasiert treffen können. Über den Parameter `control` legen Sie einen selbstgewählten Seed fest – damit stellen Sie sicher, dass das Training trotz des zufallsbasierten Verfahrens immer an der gleichen Stelle beginnt und sich Ihre Ergebnisse reproduzieren lassen.

```
fit <- LDA(dtm, k = 10, control = list(seed = 646))
```

Optional können Sie auch die Hyperparameter anpassen. Über `alpha` steuern Sie, ob sich Dokumente eher aus vielen (höheres alpha) oder wenigen Themen (niedrigeres alpha) zusammensetzen. Für einen ersten Durchlauf sind die Standardeinstellungen der Funktion – mit den Werten von 50/k für alpha und 0,1 für eta – in der Regel zielführend. Beachten Sie, dass je nach Publikation und Package unterschiedliche Bezeichnungen für diese Parameter verwendet werden. Als alternative Bezeichnung für eta findet sich häufig auch beta – nicht zu verwechseln mit der sogleich besprochenen beta-Matrix.

Das Trainieren des Modells kann eine Weile dauern. Das ermittelte Modell ist anschließend in dem Objekt `fit` abgespeichert (Abb. 8.10) und beinhaltet nun unter anderem die Wahrscheinlichkeiten, mit denen ein Wort zu einem Thema (*beta*) und mit denen ein Thema zu einem Dokument (*gamma*) gehört. Die jeweiligen Matrizen mit den Wahrscheinlichkeiten können mit der Funktion `tidy()` aus dem Objekt `fit` extrahiert und in ein Dataframe konvertiert werden. Um die Wahrscheinlichkeit auszulesen, mit der ein Wort zu einem Thema gehört, lautet der Befehl:

```
lda_woerter <- tidy(fit, matrix = "beta")
```

Tauschen Sie den Wert des Parameters `matrix` von „beta“ zu „gamma“, erhalten Sie die Themen, aus denen sich ein Dokument zusammensetzt:

```
lda_docs <- tidy(fit, matrix = "gamma")
```

Die Wahrscheinlichkeiten sind in `lda_woerter` für jede Kombination aus Wort und Thema erfasst (Abb. 8.11). Schaut man sich ein einzelnes Wort an, lässt

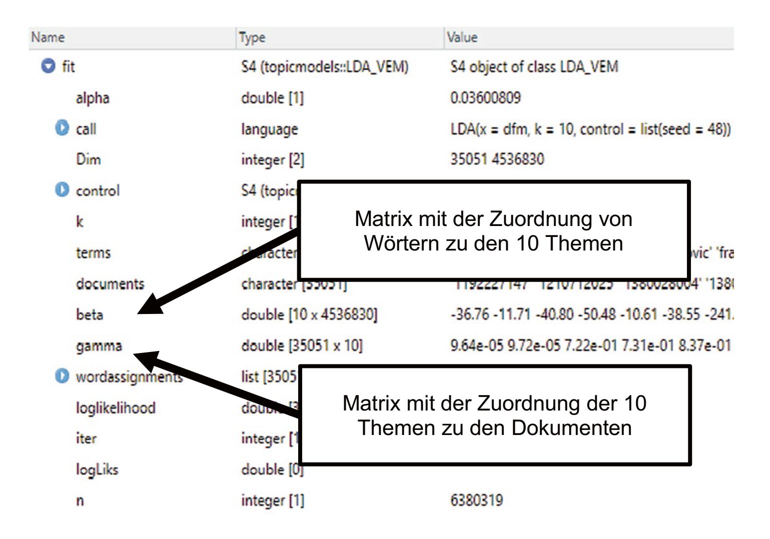


Abb. 8.10 Trainiertes Topic-Model-Objekt in R. (Quelle: eigene Darstellung)

sich ablesen, welchem Thema es am ehesten zuzurechnen ist. Das Wort „Partei" wird mit der Wahrscheinlichkeit 0,0045 am ehesten dem Topic 2 zugeschlagen, am wenigsten wahrscheinlich tritt es in Topic 4 auf.

Schritt 3: Interpretation und Validierung Topic Modeling ist ein exploratives Verfahren – Sie müssen überprüfen, ob die im ersten Versuch verwendeten Parameter dazu geeignet sind, interpretierbare Ergebnisse zu liefern, oder ob Sie das Modell nachjustieren müssen. Topic Models können sowohl interpretativ als auch durch quantitative Maßzahlen validiert werden. Für eine interpretative Herangehensweise werden die Wort-Thema- und Thema-Dokument-Zuordnungen manuell gesichtet.[26] Um die 15 in der beta-Matrix wahrscheinlichsten Wörter je Topic auszuwählen, können sie die Funktion `slice_max()` in Kombination mit `group_by()` verwenden:

[26] Hilfreich ist dafür auch das R-Package *LDAvis* (Sievert und Shirley 2014).

	topic	term	beta
1	1	partei	2.819818e-11
2	2	partei	4.540670e-03
3	3	partei	2.889160e-09
4	4	partei	5.229253e-30
5	5	partei	7.911224e-26
6	6	partei	1.052940e-03
7	7	partei	4.252028e-15
8	8	partei	1.315446e-23
9	9	partei	1.617310e-11
10	10	partei	2.344550e-16

Abb. 8.11 Wort-Thema-Wahrscheinlichkeit (beta-Werte). Basis: Auszug aus der Beta-Matrix eines Topic Models. Die Wahrscheinlichkeiten sind in wissenschaftlicher Notation angegeben. Hinter dem `e` folgt die Anzahl der Stellen, die das Komma nach links verschoben werden muss. (Quelle: eigene Darstellung)

```
top_woerter <- lda_woerter %>%
  group_by(topic) %>%
  slice_max(beta, n = 15)%>%
  ungroup()
```

Das wesentliche Kriterium besteht darin, ob sich die vom Algorithmus gefundenen Themen auch für Menschen sinnvoll interpretieren lassen. Dabei sollten Sie im Tibble `top_woerter` innerhalb eines Topic einen sinnvollen Zusammenhang zwischen den Wörtern erkennen und das jeweils dahinterliegende Thema benennen können (Abb. 8.12).

Betrachtet man nun für die drei in der Abbildung dargestellten Topics die einzelnen Wörter, lassen sich durchaus sinnvolle Themen erkennen. So sind im ersten Block vor allem Wörter rund um Politik („cdu“, „partei“, „bundestag“) prominent, das Vokabular des zweiten Blocks dreht sich um die Corona-Pandemie und der dritte Komplex bezieht sich auf Fußball („spiel“, „trainer“, „fußball“). Das Modell hat also dazu geführt, die vielen Spalten der Document-Term-Matrix auf wenige Topics zu reduzieren.

In einem zweiten Schritt sollten Sie überprüfen, ob sich die zugeordneten Themen auch tatsächlich in den Dokumenten wiederfinden. Ähnlich zu den Top-Wörtern, lassen sich die Top-Dokumente ermitteln:

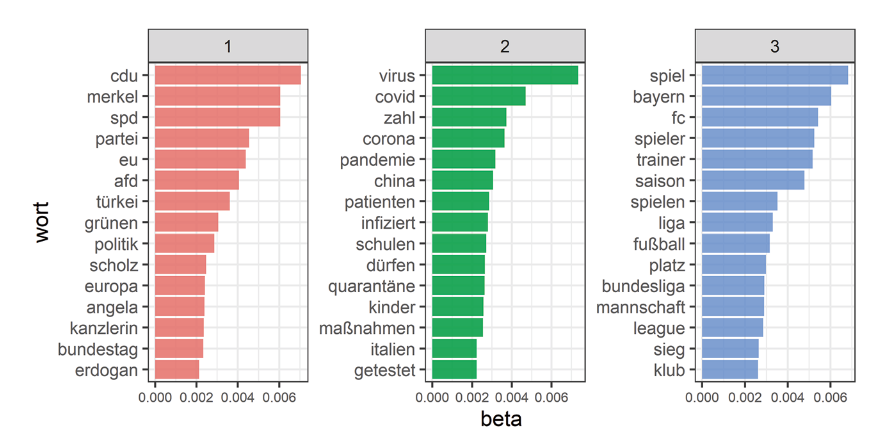


Abb. 8.12 Top-15 Wörter je Thema. Basis: Auszug aus der beta-Matrix für drei Topics. (Quelle: eigene Darstellung)

```
top_docs <- lda_topics %>%
  group_by(topic) %>%
  slice_max(gamma, n = 15) %>%
  ungroup()
```

Im Tibble `top_docs` sind lediglich die Dokumentnummern verzeichnet. Wenn Sie mit dem vorbereiteten Beispieldatensatz arbeiten, können Sie die Dokumentennummer mit der *quanteda*-Funktion `docid()` sowie Titel und URLs der Dokumente mit `docvars()` aus der Document-Feature-Matrix auslesen, in einem Tibble ablegen und schließlich an `top_docs` joinen:

```
# Liste der Dokumente aus der DFM auslesen ....
docs <- tibble(
  document = docid(dfm),
  docvars(dfm, c("title", "url"))
)

#... und an die Top-Dokumente anfügen
top_docs <- left_join(top_doc, docs, by = "document")
```

Indem man in die einzelnen Dokumente hineinliest, stellt man fest, inwiefern sich das Thema im Text wiederfindet (Abb. 8.13). So lässt sich in der Abbildung die Zuordnung des markierten Dokuments zu einem pandemiegeprägten Topic

	document	topic	gamma
1	1614611722	1	0.9998547
2	1646239479	1	0.9998489
3	1659072249	1	0.9998196
4	1537666415	2	0.9997657
5	1598747231	2	0.9997069
6	1666476432	2	0.9997840
7	1685676107	3	0.9997900
8	1692353842	3	0.9997830
9	1696350738	3	0.9997621
10	1449657559	4	0.9996958

„Kein Ergebnis in der Nacht – EU-Sondergipfel wird erneut verschoben.“

https://www.bild.de/politik/2020/politik/in-ergebnis-in-der-nacht-eu-sondergipfel-wird-erneut-verlaengert-71976554.bild.html

Abb. 8.13 Top-3 Dokumente je Thema. (Quelle: eigene Darstellung)

durchaus nachvollziehen, die Dokumente drehen sich beispielsweise um einen Sondergipfel für ein Corona-Hilfspaket der EU. Nicht nur die Spalten, auch die Zeilen der Document-Term-Matrix werden also insofern zusammengefasst, als dass sich die Dokumente zu themenspezischen Gruppen zusammenstellen lassen. Topic-Modeling leistet damit sowohl die Faktorisierung der Variablen als auch ein Clustering der Fälle.

8.2.3 Modelloptimierung

Auch wenn in den bisherigen Beispielen gut nachvollziehbare Themenzuordnungen dargestellt sind, muss man sich beim Topic-Modeling durchaus auf uneindeutige Ergebnisse einstellen. In der Regel sind einige Topics gar nicht interpretierbar, Topic Models neigen dazu, Rauschen in den Daten in nichtssagenden Topics zu sammeln. Zudem führen verschiedene Parameter schnell zu recht unterschiedlichen Zuordnungen. Um die Stabilität und Interpretierbarkeit der Modelle zu verbessern, haben sich etliche datenbasierte Verfahren etabliert, die eine interpretative Herangehensweise ergänzen. Eine statistisch optimale Modellpassung ist zwar nicht mit guter Interpretierbarkeit gleichzusetzen (DiMaggio et al. 2013, S. 582 f.), entsprechende Metriken können dennoch die eigenen Entscheidungen leiten:

- **Kohärenz**: Die Topic-Kohärenz nach Mimno et al. (2011) ist eine Maßzahl, mit der die Zusammenstellung der Wörter je Topic überprüft wird. Über die Kookkurrenz von Wörtern innerhalb eines Dokumentes wird deren semantische Nähe bestimmt. Spiegelt sich diese Nähe auch in einem Topic wieder, verfügt das Topic über einen guten Kohärenzwert.
- **Perplexity**: Über das Maß der Perplexity wird bestimmt, wie gut ein Modell generalisiert (Blei et al. 2003, S. 1008) – das heißt, wie gut es dafür geeignet ist, auch neue Daten richtig zu klassifizieren. Ähnlich wie beim überwachten maschinellen Lernen wird das Modell auf einem Großteil des Korpus trainiert und ein kleinerer Teil wird für die Validierung herangezogen. Ein niedrigerer Wert bedeutet, dass das Modell besser verallgemeinerbar ist.

Diese und andere Metriken sind zur Einschätzung der Modellgüte hilfreich, sie geben jedoch noch keine Anleitung, wie ein zuverlässiges Modell gefunden werden kann. Dazu kann man verschiedene Modelle vergleichen, die Startbedingungen des Algorithmus bewusst setzen oder das Ergebnis umgestalten:

- **Grid search**: Der LDA-Algorithmus setzt voraus, dass man die Anzahl k der Themen sowie alpha- und eta-Werte der Verteilungen vorgibt. Die geeigneten Werte für diese Parameter können systematisch bestimmt werden. Hierzu legt man unterschiedliche Wertebereiche für die Hyperparameter wie alpha, eta oder k fest und berechnet für jede Kombination ein Topic Model. Anschließend wird diejenige Parameterkombination ausgewählt, die zu den besten Kohärenzmetriken oder ähnlichen Gütekriterien führt (siehe das R-Package *ldaTuning* mit Hinweisen auf weitere Metriken; Nikita und Chaney 2020).
- **Prototyping**: Topic Models basieren auf zufallsbasierten Prozessen, jeder Durchlauf kann selbst bei gleichbleibenden Parametern zu einem anderen Modell führen. Beim Prototyping wird eine Vielzahl an Modellen gerechnet, um schließlich dasjenige Modell auszuwählen, das allen anderen am ähnlichsten ist (Rieger et al. 2020; siehe das R-Package *LDAPrototype*; Rieger 2021).
- **Seeded LDA**: Im einfachsten Fall beginnt die Modellschätzung mit einer zufälligen Zuordnung von Topics zu Wörtern. Sind bereits Informationen über die gesuchten Themen vorhanden, kann stattdessen eine Startkonfiguration vorgegeben werden, aus dem unüberwachten wird dadurch ein semiüberwachtes Lernverfahren (Lu et al. 2011).
- **Gewichtung der Relevanz**: Unter Berücksichtigung, wie häufig ein Wort im gesamten Korpus vorkommt, werden Wörter hoch- bzw. heruntergewichtet, um die Relevanz im Korpus zu erfassen (Sievert und Shirley 2014, S. 66).

Darüber hinaus finden sich vielfältige Weiterentwicklungen von Topic Models für spezifischere Anwendungsfälle bzw. Erweiterungen der Modelle auf Basis von theoretischen Annahmen (siehe Rob & Singh 2022). Dazu zählen Varianten, die sich vom Bag-of-Words-Ansatz lösen und die Reihenfolge von Wörtern berücksichtigen (Wallach 2006), die Korrelationen zwischen Topics einbeziehen (Correlated Topic Models CTM; Blei und Lafferty 2007) oder die weitere textexterne Variablen wie die Quelle der Dokumente einberechnen (Structural Topic Models STM; Roberts et al. 2016).

Übungsfragen

1. Sie haben einen Datensatz mit Kennzahlen zu Kunstwerken. Nun wollen Sie die Kunstwerke nach Epochen zusammenfassen. Wenden Sie dafür eher ein Clusterverfahren oder eher eine Faktorenanalyse an?
2. Sie haben einen annotierten Datensatz mit Zeitungsartikeln, denen die Themen „Politik", „Wirtschaft", „Sport", „Unterhaltung" zugeordnet sind. Wenden Sie ein überwachtes oder ein unüberwachtes Lernverfahren an, um automatisiert weitere Artikel nach Themen zu sortieren?
3. Topic Models sind Mixed-Membership-Modelle, die auf einem Bag-of-Words-Ansatz beruhen. Umschreiben Sie diesen Satz in Ihren eigenen Worten!
4. Ein Wort in Ihrem Korpus hat einen beta-Wert von 0,01 in Bezug auf Topic 1 und einen beta-Wert von 0,06 in Bezug auf Topic 2. Was bedeutet das?

Weiterführende Literatur

Géron, A. (2017). *Hands-On Machine Learning with Scikit-Learn & Tensorflow. Concepts, Tools and Techniques to Build Intelligent Systems.* Sebastopol: O'Reilly Media.

James, G., Witten, D., Hastie, T. & Tibshirani, R. (2013). *An Introduction to Statistical Learning. With Applications in R.* New York: Springer.

Russell, S. J. & Norvig, P. (2012). *Künstliche Intelligenz: Ein moderner Ansatz* (3. aktualisierte Aufl.). München: Pearson.

Literatur

Benoit, K., Watanabe, K., Wang, H., Nulty, P., Obeng, A., Müller, S. et al. (2018). quanteda: An R package for the quantitative analysis of textual data. *The Journal of Open Source Software*, *3*(30), 774. https://doi.org/10.21105/joss.00774

Blei, D. M. (2012). Probabilistic topic models. *Communications of the ACM*, *55*(4), 77–84. https://doi.org/10.1145/2133806.2133826

Blei, D. M. & Lafferty, J. D. (2007). A correlated topic model of Science. *The Annals of Applied Statistics*, *1*(1). https://doi.org/10.1214/07-AOAS114

Blei, D. M., Ng, A. Y. & Jordan, M. I. (2003). Latent Dirichlet Allocation. *Journal of Machine Learning Research*, *3*, 993–1022.

Bradski, G. (2000). The OpenCV library. *Dr. Dobb's Journal of Software Tools*. https://github.com/itseez/opencv

Chollet, F. (2020). Keras [Computer software]. https://keras.io/

Clark, A. & Contributors. (2022). Pillow (Version 9.2.0) [Computer software]. *Zenodo*. https://doi.org/10.5281/zenodo.6788304

DiMaggio, P., Nag, M. & Blei, D. (2013). Exploiting affinities between topic modeling and the sociological perspective on culture: Application to newspaper coverage of U.S. government arts funding. *Poetics*, *41*(6), 570–606. https://doi.org/10.1016/j.poetic.2013.08.004

Field, A., Miles, J. & Field, Z. (2012). *Discovering statistics using R*. Los Angeles: Sage.

Goodfellow, I. J., Dumitru, E., Carrier, P. L., Courville, A., Mirza, M., Hamner, B. et al. (2013). Challenges in Representation Learning: A report on three machine learning contests. *Neural Networks*, *64*, 59–63. https://doi.org/10.1016/j.neunet.2014.09.005

Google. (2022h). TensorFlow. An end-to-end machine learning platform [Computer software]. https://www.tensorflow.org/

Google. (2022i). *Tutorials. Convolutional Neural Network (CNN)*. Zugriff am 08.05.2022. https://www.tensorflow.org/tutorials/images/cnn

Griffiths, T. L. & Steyvers, M. (2004). Finding scientific topics. *Proceedings of the National Academy of Sciences*, *101*(suppl_1), 5228–5235. https://doi.org/10.1073/pnas.0307752101

Grün, B. & Hornik, K. (2011). topicmodels : An R Package for Fitting Topic Models. *Journal of Statistical Software*, *40*(13). https://doi.org/10.18637/jss.v040.i13

Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D. [David] et al. (2020). Array programming with NumPy. *Nature*, *585*(7825), 357–362. https://doi.org/10.1038/s41586-020-2649-2.

Honnibal, M. & Montani, I. (2020). spaCy: Industrial-Strength Natural Language Processing in Python (Version 3.3) [Computer software]: Explosion. https://spacy.io/

Hu, D. J. (2009). Latent Dirichlet Allocation for Text, Images, and Music. https://cseweb.ucsd.edu/~dhu/docs/research_exam09.pdf. Zugegriffen: 3. Apr. 2022.

Hunter, J., Dale, D., Firing, E., Droettboom, M. & Matplotlib development team. (2022). Matplotlib: Visualization with Python (Version 3.5.2) [Computer software]. https://matplotlib.org/

James, G., Witten, D., Hastie, T. & Tibshirani, R. (2013). *An Introduction to Statistical Learning. With Applications in R*. New York: Springer. https://doi.org/10.1007/978-1-4614-7138-7

Kaggle. (2013). *Challenges in Representation Learning: Facial Expression Recognition Challenge. Learn facial expressions from an image*. Research Prediction Competition. Zugriff am 08.05.2022. https://www.kaggle.com/c/challenges-in-representation-learning-facial-expression-recognition-challenge/leaderboard

Khaireddin, Y. & Chen, Z. (2021). *Facial Emotion Recognition: State of the Art Performance on FER2013*. Zugriff am 08.05.2021. http://arxiv.org/pdf/2105.03588v1

LeCun, Y. & Bengio, Y. (2003). Convolutional networks for images, speech, and time series. In S. Amari (Hrsg.), *The handbook of brain theory and neural networks* (2. Aufl., S. 276–278). Cambridge: MIT Press.

LeCun, Y., Bengio, Y. & Hinton, G. (2015). Deep learning. *Nature*, *521*, 436–444. https://doi.org/10.1038/nature14539

Lu, B., Ott, M., Cardie, C. & Tsou, B. K. (2011). Multi-aspect Sentiment Analysis with Topic Models. In M. Spiliopoulou (Hrsg.), *2011 IEEE 11th International Conference on Data Mining Workshops (ICDMW)* (S. 81–88). Vancouver: IEEE. https://doi.org/10.1109/ICDMW.2011.125

Maier, D., Waldherr, A. [A.], Miltner, P., Wiedemann, G., Niekler, A., Keinert, A. et al. (2018). Applying LDA Topic Modeling in Communication Research: Toward a Valid and Reliable Methodology. *Communication Methods and Measures*, *12*(2–3), 93–118. https://doi.org/10.1080/19312458.2018.1430754

McCulloch, W. S. & Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, *5*(4), 115–133. https://doi.org/10.1007/BF02478259

Mimno, D., Wallach, H., Tally, E., Leenders, M. & McCallum, A. (2011). Optimizing Semantic Coherence in Topic Models. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing* (S. 262–272). Edinburgh: Association for Computational Linguistics (ACL).

Mitchell, T. M. (1997). *Machine learning*. Boston: WCB/McGraw-Hill.

Natale, S. & Ballatore, A. (2020). Imagining the thinking machine: Technological myths and the rise of artificial intelligence. *Convergence: The International Journal of Research into New Media Technologies*, *26*(1), 3–18. https://doi.org/10.1177/1354856517715164

Neuendorf, K. A. (2017). *The content analysis guidebook* (2. Aufl.). Los Angeles: Sage.

Ng, A. Y. (coursera, Hrsg.). (2022). *Machine Learning by Stanford University*. Zugriff am 06.05.2022. https://www.coursera.org/learn/machine-learning

Nikita, M. & Chaney, N. (2020). ldatuning. Tuning of the Latent Dirichlet Allocation Models Parameters (Version 1.0.2) [Computer software]. https://cran.r-project.org/package=ldatuning

Pandas development team. (2022a). Pandas (Version 1.4.3) [Computer software]. *Zenodo*. https://doi.org/10.5281/zenodo.3509134

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O. et al. (2011). Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, *12*, 2825–2830.

Pritchard, J. K., Stephens, M. & Donnelly, P. (2000). Inference of population structure using multilocus genotype data. *Genetics*, *155*(2), 945–959. https://doi.org/10.1093/genetics/155.2.945

Puschmann, C. & Haim, M. (2021). *useNews*. [Dataset]. Zugriff am 28.01.2022. https://osf.io/uzca3/

Python Software Foundation. (2022e). os. Miscellaneous operating system interfaces (Version 3.10.5) [Computer software]. https://docs.python.org/3/library/os.html

Quanteda. (2022). *Topic Models*. Zugriff am 08.05.2022. https://tutorials.quanteda.io/machine-learning/topicmodel/

Rieger, J. (2021). ldaPrototype. Prototype of Multiple Latent Dirichlet Allocation Runs (Version 0.3.1) [Computer software]. https://cran.r-project.org/package=ldaPrototype

Rieger, J., Rahnenführer, J. & Jentsch, C. (2020). Improving Latent Dirichlet Allocation: On Reliability of the Novel Method LDAPrototype. In E. Métais, F. Meziane, H. Horacek & P. Cimiano (Hrsg.), *Natural Language Processing and Information Systems* (S. 118–125). Cham: Springer. https://doi.org/10.1007/978-3-030-51310-8_11

Rob, C. & Singh, L. (2022). The Evolution of Topic Modeling. *ACM Computing Surveys 54*(10s), 1–35. https://doi.org/10.1145/3507900

Roberts, M. E., Stewart, B. M. & Airoldi, E. M. (2016). A Model of Text for Experimentation in the Social Sciences. *Journal of the American Statistical Association*, *111*(515), 988–1003. https://doi.org/10.1080/01621459.2016.1141684

Rosenblatt, F. (1958). The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological Review*, *65*(6), 386–408. https://doi.org/10.1037/h0042519

Rumelhart, D. E., Hinton, G. E. & Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature*, *323*(6088), 533–536. https://doi.org/10.1038/323533a0

Russell, S. J. & Norvig, P. (2012). *Künstliche Intelligenz. Ein moderner Ansatz* (3., aktual. Aufl.). München: Pearson.

Sambare, M. (2020). *FER-2013. Learn facial expressions from an image*. [Dataset], kaggle.com. Zugriff am 08.05.2022. https://www.kaggle.com/datasets/msambare/fer2013

Samuel, A. L. (1959). Some studies in machine learning using the game of checkers. *IBM Journal*, *3*(3), 210–229. https://doi.org/10.1147/rd.33.0210

Scikit-learn developers. (2022). *sklearn.neural_network.MLPClassifier*. Zugriff am 08.05.2022. https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html

Sievert, C. & Shirley, K. E. (2014). LDAvis: A method for visualizing and interpreting topics. In *Proceedings of the Workshop on Interactive Language Learning, Visualization, and Interfaces* (S. 63–70). Baltimore: Association for Computational Linguistics (ACL). https://doi.org/10.13140/2.1.1394.3043

Smith, R. & Tesseract Development Team. (2022). Tesseract [Computer software]. https://github.com/tesseract-ocr

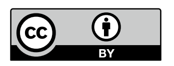
Varoquaux, G. & Joblib developers. (2021). Joblib: running Python functions as pipeline jobs (Version 1.1.0) [Computer software]. https://joblib.readthedocs.io/

Voss, C., Cammarata, N., Goh, G., Petrov, M., Schubert, L., Egan, B. et al. (2021). Visualizing Weights. *Distill*, *6*(2). https://doi.org/10.23915/distill.00024.007

Wallach, H. M. (2006). Topic modeling: beyond bag-of-words. In W. Cohen & A. Moore (Hrsg.), *Proceedings of the 23rd international conference on Machine learning – ICML '06* (S. 977–984). New York: ACM Press. https://doi.org/10.1145/1143844.1143967

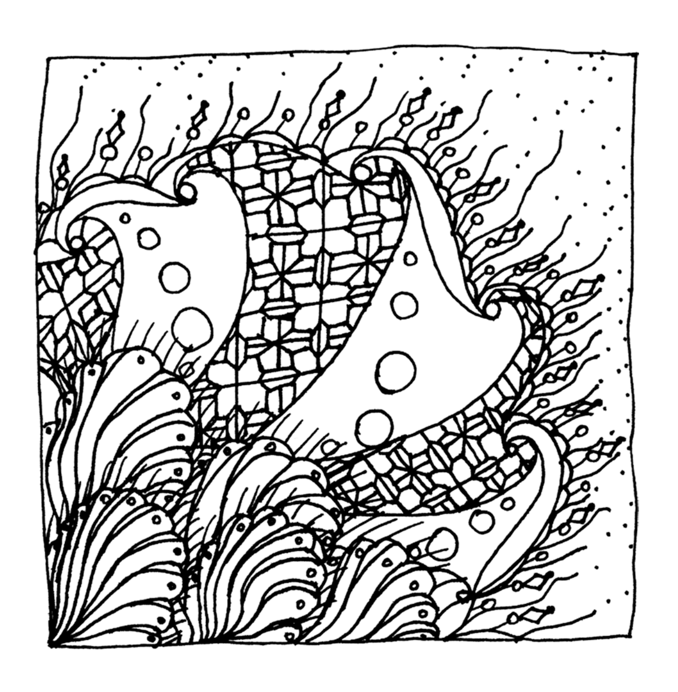
Waskom, M. L. (2021). seaborn: statistical data visualization. *The Journal of Open Source Software*, *6*(60), 3021. https://doi.org/10.21105/joss.03021

Wrobel, S., Morik, K. & Joachims, T. (2003). Kapitel 14: Maschinelles Lernen und Data Mining. In G. Görz & J. Schneeberger (Hrsg.), *Handbuch der Künstlichen Intelligenz* (S. 517–597). München: Oldenbourg. https://doi.org/10.1524/9783486598834.517


BY



9 Textanalyse



J. Jünger, C. Gärtner, *Computational Methods für die Sozial- und Geisteswissenschaften*, https://doi.org/10.1007/978-3-658-37747-2_9

Zusammenfassung

Dieses Kapitel führt in die automatisierte Textanalyse ein. Sie lernen, wie man Wortfrequenzanalysen durchführt und Texte mithilfe von Diktionären analysiert. Zudem werden grundlegende Konzepte von automatisierten Inhaltsanalysen sowie von Natural Language Processing vorgestellt.

Im Online-Repositorium unter https://github.com/strohne/cm finden Sie begleitend zum Kapitel weitere Materialien, auf die wir im Text mit ☛ verweisen.

Schlüsselwörter

Wortfrequenzanalyse · Tokenisierung · Kookkurrenz · Kollokation · Tf-idf · N-Gramme · Diktionärsbasierte Inhaltsanalyse · Sentimentanalyse · Dependenzparsing · POS-Tagging · Natural Language Parsing · Quanteda · SpaCy

Menschen nutzen Sprache, um ihrem Leben Bedeutung zu geben. In den Sozial- und Geisteswissenschaften sind sprachliche Daten deshalb ein zentraler Untersuchungsgegenstand. Dabei kann es sich um extra für wissenschaftliche Zwecke erhobene Daten wie Interviews handeln, aber auch um Dokumente wie Pressemitteilungen, Inschriften auf Grabsteinen oder Postings auf Sozialen Netzwerkseiten. Mit automatisierten Textanalysen lässt sich die Bedeutung von Sprache zwar nur auf einer formalen Ebene erfassen, das aber kann sehr hilfreich sein. So können aus Wörtern und Sätzen systematisch Erkenntnisse destilliert werden, insbesondere um sich einen Überblick über unübersichtliche oder umfangreiche Textsammlungen zu verschaffen.

In Bezug auf Text lassen sich drei verschiedene Bereiche unterscheiden, bei denen Computer eine Rolle spielen (siehe auch Krippendorff 2013, S. 213):

1. **Textanalyse:** Texte können in ihre Bestandteile zerlegt und dann deskriptiv beschrieben werden. Hierzu zählen Verfahren wie Wortfrequenzanalyse (die Häufigkeit von Wörtern wird ausgezählt), Kollokationsanalysen (das gemeinsame Auftreten von Wörtern wird ausgezählt) oder Keyword-in-Context-Analysen (KWIC; die Umgebung von Schlüsselwörtern wird angezeigt). Gemeinsam ist diesen Verfahren, dass sie im Wesentlichen auf der Ebene von Textformen arbeiten, also Wörter ohne weitere Bedeutungszuschreibung und häufig auch ohne Berücksichtigung von Syntax – damit sind die sprachlichen Beziehungen zwischen den Wörtern und Zeichen gemeint – verarbeiten. Aus

sprachwissenschaftlicher Sicht fallen diese Verfahren in den Bereich der Korpuslinguistik.
2. **Inhaltsanalyse:** Im sozialwissenschaftlichen Kontext sind häufig latente Konstrukte von Interesse, die im Zuge von Inhaltsanalysen mit manifesten Merkmalen gemessen werden (Merten 1995, S. 59). Ausgehend von Eigenschaften auf der Textoberfläche muss dann auf semantische oder pragmatische Sinngehalte geschlossen werden. Sentimentanalysen unterstellen beispielsweise, dass bestimmte Wörter als Indikatoren für positive oder negative Einstellungen stehen. Auch wenn ein Korpus auf eine geringe Menge an Themen reduziert werden soll (Clustering, Topic Modeling, siehe Abschn. 8.2), ist man auf der Suche nach den datengenerierenden Prinzipien hinter einem Text. Damit ist gemeint, dass nicht der Text selbst von Interesse ist, sondern die menschlichen Verhaltensweisen, die einen Text hervorgebracht haben oder die ein Text auslöst.
3. **Annotation & Edition:** Während bei Text- oder Inhaltsanalysen Teile der Analyse automatisiert werden, setzen insbesondere interpretative bzw. hermeneutische Verfahren auf intellektuelle Interpretation. Computer unterstützen dabei die Aufbereitung von Texten oder sind ein Hilfsmittel, um relevante Textstellen zu identifizieren, die dann manuell interpretiert werden. In den Geisteswissenschaften werden Editionen von Texten beispielsweise mit Oxygen als XML-Dokumente aufbereitet (SyncroSoft 2022). In den Sozialwissenschaften wird üblicherweise Software wie MAXQDA (VERBI Software 2021) oder ATLAS.ti (ATLAS.ti 2022) zur Organisation von Textkorpora eingesetzt. Textstellen werden markiert und verschlagwortet, diese sogenannten Codes oder Annotationen wiederum kommentiert.

In vielen Studien und Tools werden automatisierte und interpretative Verfahren kombiniert (Jünger et al. 2022). Im Folgenden geht es allerdings zunächst um die ersten beiden Punkte, das heißt um Textanalysen und Inhaltsanalysen.

9.1 Wortfrequenzanalysen

Wir lernen im Kindesalter, Sprache zu sprechen und gesprochene Sprache zu verstehen. Um gehörte Laute und gelesene Grapheme als Zeichen mit einer Bedeutung zu erkennen, ist ein jahrelanger, komplexer Lernprozess nötig. Zudem ist Sprache vielfältig und verändert sich fortlaufend, sodass die automatische Erkennung mit Computern keine leichte Aufgabe ist. Die Komplexität von Sprache kann für

maschinenbasierte Analysen reduziert werden, indem Texte zunächst als Ansammlung von Wörtern behandelt werden – man spricht dann vom Bag-of-Words-Ansatz.

Um die Anzahl von Wörtern in einem Text zu ermitteln, wird zwischen *Types* und *Token* unterschieden (Peirce 1906, S. 506). Token (auch Terms genannt) bezeichnen dabei das konkrete Auftreten von Wörtern in einem Text – also aus wie vielen Einzelwörtern ein Text besteht. Die Types geben dagegen die verschiedenen Wortarten an – also aus wie vielen unterschiedlich geschriebenen Wörtern ein Text besteht. Die Sätze „Furcht ist der Pfad zur dunklen Seite. Furcht führt zu Wut, Wut führt zu Hass, Hass führt zu unsäglichem Leid.“ bestehen beispielsweise aus 20 Token und 13 Types.[1] Ein Datenformat, mit dem Types und Token erfasst werden können und das häufig die Grundlage für automatisierte Textanalysen ist, stellt die Document-Term-Matrix (allgemein auch Document-Feature-Matrix genannt) dar. In dieser Matrix ist jedes Dokument, etwa jeder Satz oder jeder Text, in einer Zeile erfasst. Die Spalten bestehen aus den Wörtern des gesamten untersuchten Korpus. In den Zellen ist dann festgehalten, ob bzw. wie oft ein Wort in einem Dokument auftaucht (Tab. 9.1).

Dass durch die statistische Behandlung von Text Informationen verloren gehen, ist offenkundig. Im Gegensatz zum menschlichen „close reading“, bei dem die konzentrierte und schrittweise Lektüre von Äußerungen nach und nach zu einer Interpretation führt, geht es beim automatisierten „distant reading“ umgekehrt darum, aus aggregierten Verteilungen und Strukturen Sinn zu destillieren. Diese Vorgehensweise hat in der Literaturwissenschaft einen kritischen Diskurs in Gang gesetzt. Der Literaturwissenschaftler Franco Moretti, der diese Methoden populär gemacht hat, beschreibt eine solche Form der Analyse als „a little pact with the

Tab. 9.1 Eine Document-Term-Matrix

doc_id	der	dunklen	führt	furcht	hass	ist	leid	pfad	seite	unsäglichem	wut	zu	zur
Satz 1	1	1	0	1	0	1	0	1	1	0	0	0	1
Satz 2	0	0	3	1	2	0	1	0	0	1	2	3	0

Quelle: eigene Darstellung

[1] Alternativ kann man auch zwölf Types zählen, wenn man die Ausprägungen „zu“ und „zur“ als ein Wort betrachtet.

devil: we know how to read texts, now let's learn how not to read them" (Moretti 2013, S. 48). Sprachliche Äußerungen werden teilweise so in Einzelteile zerlegt, dass die Reihenfolge und die Zusammenhänge zwischen den Wörtern verloren gehen. Eine Filmkritik wie „nicht gut, sehr langweilig" wird damit in die Wörter „nicht", „gut", „sehr" und „langweilig" zerteilt. Betrachtet man nur das reine Vorkommen der Wörter, dann ist dies gleichwertig zu „nicht", „langweilig", „sehr" und „gut". Zählt man die Adjektive in diesen Texten aus, würde die Anzahl für „gut" und für „langweilig" in beiden Fällen gleich sein.

Trotz des Informationsverlustes, oder vielleicht sogar gerade wegen der Konzentration auf die wesentlichen bedeutungstragenden Teile eines Textes, lassen sich allein mit Wortfrequenzanalysen durchaus hilfreiche Einsichten gewinnen. So kann man sich einen schnellen Überblick über umfangreiche Korpora verschaffen. Für diese Analysen gibt es spezialisierte Programme wie AntConc (Anthony 2022) und Sketch Engine (Lexical Computing 2022). Sie lassen sich aber auch mit Statistiksoftware wie R oder Python (siehe Kap. 5) umsetzen. Während auf Textanalyse zugeschnittene Programme einen guten Einstieg erlauben und durch die Benutzeroberflächen typische Analysen vorgeben, gewinnt man durch Programmiersprachen eine hohe Flexibilität, muss dafür aber jede einzelne Entscheidung bei der Textverarbeitung selbst treffen. Das folgende R-Skript verdeutlicht zunächst einige typische Schritte, die in dieser oder ähnlicher Form am Beginn jeder Textanalyse stehen.

Schritt 1: Texte einlesen Im Gegensatz zu fertigen Datensätzen liegen Textkorpora häufig als einzelne Textdateien oder auch als Word- oder PDF-Dateien vor. Diese Dateien lassen sich mit der Funktion `readtext()` aus dem gleichnamigen Package einlesen (Benoit et al. 2021). Angenommen alle Dateien liegen als UTF8-kodierte Textdateien (siehe Abschn. 3.2) vom Projektverzeichnis aus gesehen im Unterordner *korpus*, dann können sie wie folgt eingelesen werden (☛ *Repositorium*):[2]

```
library(readtext)
texte <- readtext("korpus", encoding = "UTF-8")
```

Das Ergebnis ist eine Tabelle mit den Dateinamen in der ersten Spalte und dem Inhalt der jeweiligen Datei in der zweiten Spalte (Tab. 9.2). Schauen Sie sich für

[2] Das Beispielkorpus umfasst 77 kurze Texte von Studierenden zu der Frage, wie sie den Umgang mit Daten durch Online-Unternehmen einschätzen.

Tab. 9.2 Ausschnitt des eingelesenen Textkorpus in RStudio

doc_id	text
RF1.txt	Die Angabe, die am häufigsten zu Verwendung der weite…
RF10.txt	Bei der Nutzung vieler Internetdienste werden private…
RF11.txt	Die Nutzung digitaler Internetplattformen ist heutzut…
RF12.txt	Seit der neuen Datenschutzordnung fällt insbesondere…
RF13.txt	Die Nutzung von Suchmaschinen wie Google oder Unt…
RF14.txt	Die persönlichen Daten werden von den jeweiligen Plat…
RF15.txt	Durch die Nutzung von Internetdiensten werden viele D…
RF16.txt	Ich denke, dass die bei Google, Facebook & Co. anfall…
RF17.txt	Die Datenverbreitung im Internet findet im Hintergund…

Quelle: Eigene Darstellung basierend auf dem Beispielkorpus (☛ *Repositorium*)

diese und die im Folgenden verwendeten Funktionen die Hilfe an.[3] Es gibt zahlreiche Optionen für vielfältige Eventualitäten.

Schritt 2: Texte tokenisieren Im nächsten Schritt werden die Texte in einzelne Sätze, Wörter oder andere Einheiten zerlegt. Die einzelnen Einheiten werden Token genannt, der Vorgang entsprechend Tokenisierung. Eine gute Wahl für erste eigene Versuche ist die Funktion `unnest_token()` aus dem Package *tidytext* (Silge und Robinson 2016). Der erste Parameter gibt den Dataframe mit den Texten an. Der zweite Parameter bestimmt den Namen der Output-Variable, das heißt, wie die Spalte mit den einzelnen Wörtern heißen soll. Der dritte Parameter bestimmt schließlich den Namen der Input-Spalte, die den zu zerlegenden Text enthält:

```
library(tidytext)
woerter <- unnest_tokens(texte, wort, text)
```

Aus dem oben angeführten Beispiel wird eine neue Tabelle erstellt. Der Dateiname bleibt in der ersten Spalte `doc_id` erhalten, in der zweiten Spalte `wort` werden untereinander die einzelnen Wörter der entsprechenden Datei aufgeführt (Tab. 9.3).

Diese Funktion kann auch dazu genutzt werden, den Text in größere Einheiten wie Sätze sowie in kleinere Einheiten wie Zeichen zu zerlegen. Dafür wird

[3] Platzieren Sie den Cursor auf dem Funktionsnamen und drücken Sie die Taste F1. Oder stellen Sie vor den Befehl ein Fragezeichen und führen Sie den Befehl aus, zum Beispiel `?readtext`.

Tab. 9.3 Ausschnitt des Outputs nach der Tokenisierung von Texten in Wörter

doc_id	wort
RF1.txt	die
RF1.txt	angabe
RF1.txt	die
RF1.txt	am
RF1.txt	häufigsten
RF1.txt	zu
RF1.txt	verwendung
RF1.txt	der
RF1.txt	weitergegebenen

Quelle: Eigene Darstellung

innerhalb der Funktion der Parameter `token` verwendet (zum Beispiel `token="sentence"`); voreingestellt ist dieser Parameter auf `token="words"`.[4]

Schritt 3: Texte bereinigen Im Zusammenhang mit der Tokenisierung werden die Daten häufig bereinigt. Typischerweise zählen dazu folgende Schritte, die aber an die konkrete Fragestellung angepasst werden müssen:

- **Groß- oder Kleinschreibung:** Durch die Funktion `unnest_token()` werden alle Wörter in Kleinschreibung umgewandelt. Das ist in der Regel sinnvoll, da etwa Adjektive am Satzanfang großgeschrieben werden, ohne dass dies einen inhaltlichen Unterschied zu kleingeschriebenen Wörtern innerhalb eines Satzes darstellt. Für ein Computerprogramm würde es sich sonst um zwei unterschiedliche Zeichenfolgen handeln.

  Mit dem Parameter `to_lower=F` kann die Voreinstellung verändert werden. Falls die Daten erst später in Kleinschreibung umgewandelt werden sollen, hilft die Funktion `str_to_lower()` aus dem Package *stringr* weiter (Wickham 2019a). Das folgende Beispiel bettet diese Funktion in ein typisches Tidyverse-Muster mit Zuweisungsoperator `<-`, Pipe `%>%` und `mutate()`-Funktion ein (siehe Abschn. 5.1):

[4] Voreingestellte Parameter (engl. *default* values) müssen nicht extra angegeben werden, lediglich wenn Sie diese abändern wollen. Welche Parameter in einer Funktion bereits mit einer Voreinstellung belegt sind, entnehmen Sie der Hilfe.

```
woerter <- woerter %>%
  mutate(wort = str_to_lower(wort, locale = "de"))
```

Die Angabe `locale="de"` stellt zusätzlich sicher, dass Umwandlungskonventionen an der deutschen Sprache orientiert sind.

- **Zahlen und Sonderzeichen**: Auch die anderen Funktionen aus dem Package *stringr* sind sehr hilfreich für die Aufbereitung von Texten. Zahlen und Sonderzeichen lassen sich bei Bedarf mit regulären Ausdrücken (siehe Abschn. 4.1.1) entfernen. Der Ausdruck `[0-9]+` würde zum Beispiel alle Zahlen finden oder der Ausdruck `^[a-zäöüß]+$` alle Zeichenketten, die von Anfang (Caret `^`) bis Ende (Dollar `$`) ausschließlich aus den angegebenen Kleinbuchstaben bestehen. In Kombination mit den Funktionen `filter()` zum Filtern von Zeilen und `str_detect()` zum Entdecken der Zeichenketten lassen sich mit diesem Ausdruck alle Wörter mit Sonderzeichen entfernen (bzw. diejenigen ohne Sonderzeichen behalten):

```
woerter <- woerter %>%
  filter(str_detect(wort, "^[a-zäüöß]+$"))
```

- **Stoppwörter**: Die häufigsten Wörter in einem Text sind häufig am wenigsten interessant. Dazu zählen vor allem Artikel wie „der“ oder „ein“. Diese für eine Analyse unbedeutenden Wörter werden Stoppwörter genannt und in der Regel vor der Auswertung entfernt. Eine Liste typischer deutscher Stoppwörter liefert die Funktion `stopwords()` aus dem gleichnamigen Package (Benoit, Muhr und Watanabe 2021):

```
library(stopwords)
stopwords("de")
```

Die ersten neun Stoppwörter in dieser Liste lauten:

```
[1] "aber"      "alle"      "allem"
[4] "allen"     "aller"     "alles"
[7] "als"       "also"      "am"
```

Eine Möglichkeit zum Abgleich der Wörtertabelle mit einer solchen Liste besteht darin, die Liste der Stoppwörter zunächst in ein Dataframe bzw. Tibble umzuformen:

```
stoppwoerter <- tibble(wort = stopwords("de"))
```

Anschließend kann ein `anti_join()`[5] verwendet werden, um nur diejenigen Datensätze zu behalten, die nicht in der Stoppworttabelle enthalten sind. Im Beispiel wird die Spalte `wort` in beiden Tabellen so abgeglichen, dass in der Wörtertabelle nur Zeilen beibehalten werden, die nicht in der Stoppworttabelle zu finden sind:

```
woerter <- woerter %>%
  anti_join(stoppwoerter, by = "wort")
```

Diese Technik kann auch verwendet werden, um eine eigene Liste mit Stoppwörtern einzubinden. Dazu können Sie die Stoppwörter als CSV-Datei anlegen, in R einlesen und mit dem tokenisierten Datensatz abgleichen.

- **Stemming und Lemmatisierung**: Wenn man nicht an den konkreten Wortformen interessiert ist, sondern nur an den semantischen Dimensionen, wird man die Wörter in der Regel auf ihren Wortstamm reduzieren. Dieses sogenannte Stemming entfernt grammatisch bedingte Endungen. So wird aus den Wörtern „Hauses" und „Häusern" das Wort „Haus" oder aus den Verben „läuft" und „laufen" wird „lauf". Das Package *SnowballC* (Bouchet-Valat 2020) liefert eine entsprechende Funktion für die deutsche Sprache:

```
library(SnowballC)
woerter <- woerter %>%
  mutate(stem = wordStem(wort, language = "de"))
```

Anstelle von Stemming wird mitunter eine Lemmatisierung vorgenommen. Der Unterschied besteht darin, dass alle Wörter nicht auf den Stamm, sondern auf die Grundform zurückgeführt werden. So wird aus „läuft" der Infinitiv „laufen", was sich bei der Ausgabe leichter interpretieren lässt.

- **Relative pruning:** Statt festgelegte Stoppwörter zu entfernen, können die häufigsten und seltensten Wörter entfernt werden und zum Beispiel nur Wörter beibehalten werden, die in mindestens 1 % und maximal 99 % der Texte vorkommen. Insbesondere wenn statistische Modelle (z. B. Topic Modeling, siehe Abschn. 8.2) erstellt werden, ist dieser Schritt sinnvoll. Denn Wörter, die in sehr vielen Texten

[5] Siehe Abschn. 4.2.2 für unterschiedliche Möglichkeiten, Daten über Joins zusammenzuführen.

vorkommen, können nicht zur Unterscheidung der Texte verwendet werden. Wenn Modelle darauf aufbauen, dass Wörter gemeinsam in einem Dokument auftreten (Kookkurrenz, siehe unten), sollten auch die sehr seltenen Wörter entfernt werden, denn sie eignen sich nicht gut zur Schätzung typischer Kombinationen. Zum Bestimmen der häufigen und seltenen Wörter können die im Folgenden beschriebenen Verfahren zum Auszählen der Wörter verwendet werden.

Mit diesen fünf Aufbereitungsschritten gehen einerseits Informationen verloren. Andererseits wird der Text bereits auf wesentliche Elemente reduziert. Weitere teils aufwendigere Maßnahmen zur Textaufbereitung umfassen die Auflösung von Synonymen oder die Desambiguierung gleichlautender Wörter. Auch eine Kategorisierung von Wortarten ist mitunter angebracht (POS-Tagging, siehe unten).

Schritt 4: Wörter auszählen Ist ein Text erst einmal in die Form eines Datensatzes überführt und aufbereitet worden, kann die Auswertung mit statistischen Funktionen beginnen. Am einfachsten ist das Auszählen von Worthäufigkeiten:

```
woerter %>%
  count(wort, sort = T)
```

Der Parameter `sort=T` sorgt dafür, dass die häufigsten Wörter zuerst ausgegeben werden (Tab. 9.4).

Da es sich hierbei um einen Dataframe handelt, kann das Ergebnis in einem neuen Objekt abgelegt und bei Bedarf als CSV-Datei exportiert werden. Das Ergebnis kann auch für Visualisierungen verwendet werden. Welche Wörter in

Tab. 9.4 Die häufigsten Wörter im Beispielkorpus

wort	n
daten	347
dass	167
werbung	90
internet	78
nutzer	62
dritte	58
weitergegeben	48
google	45
jedoch	43
youtube	43

Quelle: Eigene Darstellung

einem Korpus besonders häufig auftauchen, lässt sich schnell in Wordclouds erkennen. Das Package *ggwordcloud* (Le Pennec und Slowikowski 2019) ist eine Erweiterung des Visualisierungspackages *ggplot2*, mit dem solche Wordclouds in R erstellt werden können (siehe Abschn. 5.1.5). Damit die Grafik nicht mit allen Wörtern des Korpus überladen wird, lohnt es sich, nur die häufigsten Wörter abzubilden. Mit der Funktion `slice_head()` lässt sich der Output der `count()`-Funktion auf die oberen Zeilen einschränken – beispielsweise durch die Angabe `n = 50` auf die Top-50 der häufigsten Wörter:

```
topterms <- woerter %>%
  count(wort, sort = T) %>%
  slice_head(n = 50)
```

Die Wordcloud kann anschließend als *ggplot2*-Grafik erstellt werden. Dabei werden, wie bei *ggplot2* üblich, zunächst über die Funktion `aes()` die Daten den visuellen Merkmalen zugeordnet. Die Wörter werden als `label` übergeben. Der optionale Parameter `size` legt fest, wovon die Größe der Wörter in der Wordcloud abhängen soll – im Beispiel von den ausgezählten Häufigkeiten `n`, sodass häufigere Wörter größer dargestellt werden als seltenere Wörter. Über den Parameter `color` lässt sich die Farbe zuordnen. An diese Ausgangsdefinition werden mit einem + weitere Schichten der Grafik angehängt. Die Verwandlung in eine Wordcloud übernimmt die Funktion `geom_text_wordcloud()`. Es folgen weitere Formatierungen, im Beispiel die maximale Größe der Wörter und das Theme mit einer Angabe der Basisschriftgröße:

```
topterms %>%

  ggplot(aes(label = wort, size = n, color = n)) +
  geom_text_wordcloud() +

  scale_size_area(max_size = 15) +
  theme_bw(base_size = 10)
```

Die Wordcloud wird in RStudio ausgegeben und kann mithilfe der Funktion `ggsave()` abgespeichert werden (Abb. 9.1).

Schritt 5: Kookkurrenz Eine einfache Häufigkeitsanalyse ist vor allem vergleichend interessant, etwa um in der Berichterstattung den Wandel der Themen und des Sprachgebrauchs über Jahrzehnte zu verfolgen oder wenn man die Öffentlichkeitsarbeit von Organisationen (PR) der publizistischen Berichterstattung (Journalismus) gegenüberstellen will. Aufschlussreich ist aber nicht nur der Vergleich ver-



Abb. 9.1 Wordcloud mit den häufigsten 50 Wörtern des Beispielkorpus. (Quelle: eigene Darstellung)

schiedener Dokumente untereinander, sondern auch der Zusammenhang von Wörtern innerhalb der Texte. Eine einfache Analysemöglichkeit besteht darin, das gemeinsame Auftreten von Wörtern in einem Dokument auszuzählen (Kookkurrenz).[6] Nach der Tokenisierung liegt zunächst eine lange Tabelle vor, in der in einer Spalte `doc_id` der Dateiname des Statements und in einer anderen Spalte `wort` das Wort steht. Nun kann mit der Funktion `pairwise_count()` aus dem *widyr*-Package ausgezählt werden, in wie vielen Dokumenten verschiedene Wörter gemeinsam vorkommen (Robinson et al. 2021):[7]

```
library(widyr)
kookkurrenz <- woerter %>%
  pairwise_count(wort, doc_id)
```

Sortiert man das Ergebnis nach der Häufigkeit, offenbaren sich typische Zusammenhänge. Im Beispiel (Tab. 9.5) zeigt sich, dass die Begriffe „daten“ und „werbung“ sehr häufig zusammen auftreten.

Diese reine Häufigkeitsauszählung sagt noch wenig darüber aus, ob die häufigen Wörter oder die gefundenen Kookkurrenzen in dem Sinne bedeutsam sind, dass sie häufiger sind als typischerweise zu erwarten wäre. Das lässt sich hier gut daran erkennen, dass Wörter wie „dass“ oder „jedoch“ sehr häufig in Kombination mit anderen Wörter vorkommen – die verwendete Stoppwortliste ist im Beispiel nicht ideal.[8] So ist es wahrscheinlich, dass auch im alltäglichen Sprachgebrauch das Wort „dass“ häufig verwendet wird und dementsprechend auch häufig mit dem

[6] Auf dieser Grundlage lassen sich auch Netzwerke erstellen, siehe Kap. 10.

[7] Die Anzahl wird durch Matrixmultiplikation berechnet, siehe Abschn. 4.2.

[8] Passen Sie das Skript so an, dass diese Wörter bei der Aufbereitung entfernt werden!

Tab. 9.5 Kookkurrenz von Wörtern im Beispielkorpus (Auszug)

item1	item2	n
dass	daten	61
daten	dass	61
werbung	daten	45
daten	werbung	45
dritte	daten	42
daten	dritte	42
internet	daten	38
werbung	dass	38
dass	werbung	38
daten	internet	38
nutzer	daten	37
daten	nutzer	37
weitergegeben	daten	35
nutzer	dass	35
daten	weitergegeben	35
dass	nutzer	35
jedoch	daten	34
dritte	dass	34
daten	jedoch	34

Quelle: Eigene Darstellung

Wort „daten“ zusammen auftritt – ohne dass dies direkt auf eine Anomalie hindeutet. Neben inferenzstatistischen Verfahren, bei denen die statistische Signifikanz geprüft wird, lässt sich die Auffälligkeit von einzelnen Wörtern mittels Tf-idf-Werten und die Bedeutsamkeit von Kookkurrenzen mithilfe von PMI-Werten berechnen.

Das Tf-idf-Maß gibt an, wie spezifisch ein Wort für jedes einzelne Dokument ist, sodass sich je Dokument die im Vergleich zu allen anderen Dokumenten auffälligen Wörter bestimmen lassen. Der Wert setzt sich aus der Häufigkeit eines Wortes in einem Dokument (tf = term frequency) und der Häufigkeit aller Dokumente mit diesem Wort (df = document frequency) zusammen. Je höher die Term Frequency ist, umso häufiger kommt das Wort in einem Dokument vor. Ein Wert von 0,06 für das Wort „anzeigen“ in Dokument 31 besagt, dass dieses Wort 6 % aller Wörter in diesem Dokument ausmacht (Tab. 9.6). Die Document Frequeny gibt an, in wie vielen Dokumenten des Korpus das Wort enthalten ist und wird durch Logarithmieren und Invertieren (idf = inverse document frequency) zu einem Gewichtungsfaktor umgerechnet, der dann mit der Term Frequency multipliziert

Tab. 9.6 Beispiele für Tf-idf-Werte in einem Beispielkorpus zum Thema Daten

wort	doc_id	n	tf	idf	tf_idf
anzeigen	RF31.txt	7	0,062	2,958	0,183
empfänger	RF38.txt	2	0,039	4,344	0,170
gezielte	RF35.txt	3	0,046	3,651	0,168
verwendung	RF47.txt	3	0,052	2,958	0,153
evtl	RF47.txt	2	0,034	4,344	0,150
datenweitergabe	RF47.txt	3	0,052	2,734	0,141
zielgruppenorientierte	RF16.txt	2	0,030	4,344	0,132
finde	RF3.txt	3	0,045	2,734	0,124
versicherung	RF6.txt	2	0,029	4,344	0,124
erlaubt	RF47.txt	4	0,069	1,705	0,118
kaufverhalten	RF38.txt	2	0,039	2,958	0,116
minimal	RF41.txt	3	0,026	4,344	0,114
konto	RF22.txt	3	0,026	4,344	0,112

Die Tf-idf-Werte sind absteigend sortiert. (Quelle: Eigene Darstellung)

wird.[9] Je höher dieser Gewichtungsfaktor ist, desto seltener ist das Wort in allen Dokumenten. Dadurch werden die seltenen Wörter hochgewichtet und die sehr häufigen Wörter heruntergewichtet. Das daraus hervorgehende Tf-idf-Maß ist dann umso höher, je spezifischer das Wort für das Dokument ist.

Auf diese Weise lassen sich in den einzelnen Dokumenten die jeweils bedeutsamen Wörter bestimmen. Ein hoher Tf-idf-Wert für das Wort „datenweitergabe" in Text 47 würde besagen, dass dieser Text sich anders als die anderen Texte besonders mit diesem Aspekt beschäftigt. Gleichzeitig erhalten gängige und unspezifische Wörter wie „dass" oder „jedoch" einen geringen Wert, weil sie in einer Vielzahl von Dokumenten vorkommen. Darüber lassen sich also auch nach der Datenaufbereitung weitere korpusspezifische Stoppwörter identifizieren und ggf. ausfiltern. Interessant ist eine solche Berechnung vor allem dann, wenn man verschiedene Artikel vergleichen oder besonders relevante Artikel identifizieren will.

In R hilft das Package *tidytext* bei der Berechnung von Tf-idf. Grundlage ist zunächst, dass man die Anzahl der Wörter je Dokument auszählt, dann kann die Funktion `bind_tf_idf()` daraus die entsprechenden Werte berechnen:

```
tfidf <- woerter %>%
  count(wort, doc_id) %>%
  bind_tf_idf(wort, doc_id, n)
```

[9] Eine Erläuterung der Berechnung findet sich beispielsweise in Silge und Robinson (2017), Kap. 3.

Tf-idf quantifiziert damit die Beziehung zwischen Dokumenten und Wörtern. In Bezug auf auffällige Kookkurrenzen, das heißt die Beziehung zwischen zwei Wörtern, ist dagegen die Berechnung von Pointwise Mutual Information (PMI) hilfreich (siehe auch Bouma 2009). Ein hoher PMI-Wert besagt, dass zwei Wörter häufiger gemeinsam auftreten als zu erwarten wäre. Auch hierfür wird das gesamte Korpus als Vergleichsmaßstab herangezogen. Nehmen wir an, ein Korpus besteht aus 5000 Wörtern, das Wort „daten" kommt darin 100-mal vor und das Wort „computer" 50-mal. Dann beträgt die Auftretenswahrscheinlichkeit für „daten" 100/5000 = 0,02. Die Auftretenswahrscheinlichkeit für „computer" liegt bei 50/5000 = 0,01. Die Wahrscheinlichkeit, dass beide Wörter gemeinsam auftreten, kann dann durch Multiplikation berechnet werden und beträgt also 0,02 × 0,01 = 0,0002. Kommt diese Kombination nun aber tatsächlich häufiger vor, so ist sie potenziell bedeutsam. Um die Bedeutsamkeit zu quantifizieren, wird die tatsächliche zur erwarteten Wahrscheinlichkeit ins Verhältnis gesetzt. Durch Logarithmieren wird das Verhältnis 1,0 (wenn die Wahrscheinlichkeiten gleich sind) zu einem PMI von 0, Verhältnisse darunter werden zu negativen und darüber zu positiven Werten.

Die PMI kann zum einen auf Grundlage von Kookkurrenz in Dokumenten berechnet werden, aber auch auf Grundlage von Wortkombinationen. Solche Wortkombinationen werden als N-Gramme bezeichnet, wobei das N die Anzahl der Wörter angibt. Ein Unigram oder 1-Gram bezeichnet einzelne Wörter, ein Bigram oder 2-Gram die Abfolge von zwei Wörtern und ein Trigram bzw. 3-Gram die Sequenz aus drei Wörtern.[10] Bei der Tokenisierung von Text lassen sich demnach nicht nur Unigramme erzeugen, sodass jedes Wort getrennt wird, sondern auch andere N-Gramme. Auf diese Weise geraten die syntaktischen Zusammenhänge zwischen Wörtern etwas stärker in den Blick; der Bag-of-Words-Ansatz wird gelockert, indem Wortkombinationen analysiert werden.

Wie bei der Kookkurrenz von Wörtern in einem Dokument lässt sich berechnen, inwiefern Bigramme häufiger vorkommen als zu erwarten wäre. Eine solchermaßen signifikante Wortkombination wird auch Kollokation genannt. Bei der praktischen Umsetzung kann man zweistufig vorgehen und zunächst die Bigramme extrahieren und nummerieren. Anschließend werden die Bigramme wieder in einzelne Wörter zerlegt, um die Kookkurenz der Wörter innerhalb der Bigramme auszuzählen:

[10] Wenn nicht unmittelbar zusammenhängende Wörter, sondern der weitere Kontext betrachtet werden soll, können auch Skip-Gramme eingesetzt werden. Dabei werden während der Tokenisierung Wörter übersprungen (engl. *skipped*). Neben der Angabe, um welches N-Gram es sich handelt, wird über den Parameter K angegeben, wie viele Wörter maximal übersprungen werden dürfen. Der Anfang des vorigen Satzes kann beispielsweise mit einem K von 3 in die Skip-Gramme „Neben der", „Neben Angabe", „Neben um" sowie „Neben welches" unterteilt werden.

```
# Bigramme extrahieren
bigrams <- texte %>%
  unnest_tokens(bigram, text, token = "ngrams", n = 2)

# Kollokationen zählen
bigrams <- bigrams %>%
  mutate(bigram_no = row_number()) %>%
  unnest_tokens(wort, bigram) %>%
  pairwise_count(wort, bigram_no)
```

Daraus lassen sich dann der Anteil des Bigrams am Korpus, der Anteil der einzelnen Wörter an allen Wörtern und schließlich die Pointwise Mutual Information berechnen. Das tatsächliche Auftreten eines Bigrams wird wieder ins Verhältnis zum zufällig erwartbaren Auftreten gesetzt, welches auf Grundlage geschätzter Einzelwahrscheinlichkeiten der Wörter berechnet wird. Daraus wird durch Logarithmieren die PMI gewonnen. Negative Werte weisen auf Kombinationen hin, die seltener als erwartbar auftreten, und positive Werte zeigen überzufällig häufige Kombinationen an:

```
bigrams <- bigrams %>%

  # Auftretenswahrscheinlichkeit p des Bigrams
  mutate(p = n / sum(n)) %>%

  # Auftretenswahrscheinlichkeit p1 für Wort 1
  group_by(item1) %>%
  mutate(n1 = sum(n)) %>%
  ungroup() %>%
  mutate(p1 = n1 / sum(n)) %>%

  # Auftretenswahrscheinlichkeit p2 für Wort 2
  group_by(item2) %>%
  mutate(n2 = sum(n)) %>%
  ungroup() %>%
  mutate(p2 = n2 / sum(n)) %>%

  # Berechnung von pmi
  mutate(pmi = log(p / (p1 * p2)))
```

Tab. 9.7 Beispiele für Bigramme in einem Beispielkorpus zum Thema Daten

item1	item2	n	p	n1	p1	n2	p2	pmi
kritischen	bereich	1	3,2E-05	1	3,2E-05	2	6,5E-05	9,64
vereinfachen	datenklau	1	3,2E-05	1	3,2E-05	2	6,5E-05	9,64
…								
falsche	hände	3	9,72E-05	8	0,0003	6	0,0002	7,56
…								
werden	benutzt	5	1,62E-05	575	0,0186	16	0,00052	2,82

Die PMI-Werte sind absteigend sortiert. Quelle: Eigene Darstellung

Im hier verwendeten Beispielkorpus treten die Wörter „bereich“ und „kritischen“ jeweils nur ein- bzw. zweimal innerhalb eines Bigrams auf. Dass ausgerechnet diese beiden Wörter dann in einer Kombination auftreten, wäre bei vollständiger Unabhängigkeit nicht zu erwarten und führt deshalb zu einem hohen PMI (Tab. 9.7). Dennoch muss man bei der Interpretation von solch seltenen Kombinationen zurückhaltend sein – eine Aussage, die auf lediglich einer Kombination beruht, ist kaum belastbar. Solche Einzelfallkombinationen kommen prinzipbedingt durchaus oft vor, im Beispielkorpus finden sich über 18.000 singuläre Bigramme. Das rührt daher, dass die meisten Wörter in einem Korpus eher selten vorkommen[11] und jedes Wort aber natürlich auch immer in mindestens einer Kombination steht. Erst wenn man sich in höheren Bereichen umschaut – etwa Bigramme, die mindestens drei-, fünf- oder zehnmal vorkommen – gewinnt man eine etwas bessere Interpretationsgrundlage.

Eine noch robustere Einschätzung liefern Zusammenhangsmaße, bei denen solche Einzelfälle auf Grundlage weiterer statistischer Überlegungen zum Informationsgehalt von seltenen Wörtern und Kombinationen aussortiert werden. In der Computerlinguistik verbreitet ist das Log-Likelihood-Ratio (LLR, auch G2 genannt; siehe Dunning 1993), mit dem der Überraschungswert einer Kookkurrenz berechnet wird (☛ *Repositorium*). Hierdurch lassen sich etwa Kombinationen wie „daten verwendet“, „daten verkauft“ oder „jedoch weitergegeben“ finden, die durchaus typisch für den untersuchten Anwendungsbereich erscheinen.

Die drei angesprochenen Werte – Tf-idf, PMI und LLR – sind vergleichsweise einfache Verfahren, um sich einen Überblick über Wörter und Kombinationen in einem Text zu verschaffen. Grundlegend ist stets, dass das Vorkommen von Wörtern in Texten ausgezählt wird. Zu beachten ist, dass weder häufige Wörter noch

[11] Das sogenannte Zipf’sche Gesetz beschreibt eine solche Verteilung, bei der wenige Wörter sehr häufig und viele Wörter sehr selten vorkommen, zu solchen Verteilungen siehe zum Beispiel Newman (2005) und Piantadosi (2014).

seltene Kombinationen automatisch bedeutsam sind, sondern eine statistische und interpretative Einordnung vorgenommen werden muss.

Schritt 6: Keywords-in-Context Für weitere interpretative Schritte lohnt sich ein Blick in das Package *quanteda* (Benoit et al. 2022). Die vorher mit `readtext()` oder anderen Verfahren eingelesenen Texte werden in ein Korpus-Objekt umgewandelt und können dann mit vorbereiteten Auswertungsfunktionen analysiert werden. Die Funktion `kwic()` (= keywords in context) zeigt beispielsweise den Kotext[12] zu ausgewählten Schlüsselwörtern an, das heißt vorangegangene und folgende Wörter. Das folgende Beispiel gibt die Kotexte des Worts „Daten“ aus:

```
# Dateien aus dem Ordner korpus laden
texte <- readtext("korpus", encoding = "UTF-8")

# In quanteda-Corpus umwandeln
texte <- corpus(texte)

# Überblick über das Korpus
summary(texte)

# Tokenisierung
token <- tokens(texte)

# Keywords-in-Context zum Wort "daten"
kwic_daten <- kwic(token, "daten")
```

Damit wird ein Dataframe erzeugt, der in RStudio als durchsuchbare Liste aufbereitet wird und so einen schnellen Überblick über den Kontext von ausgewählten Schlüsselwörtern erlaubt (Tab. 9.8).

Das Package *quanteda* ebnet darüber hinaus mit einer Vielzahl von weiteren Funktionen den Weg zu elaborierten Verfahren automatischer Textanalyse, etwa wenn Topic Modeling zum Extrahieren der Themenstruktur eines Korpus eingesetzt werden soll (siehe Abschn. 8.2), und auch zu Visualisierungstechniken wie Wordclouds. Für eine Einführung in *quanteda* ist das Quickstart-Tutorial des Packages empfehlenswert.[13]

[12] Kotexte sind die Kontexte von Wörtern.

[13] Siehe Benoit et al. (2022; https://quanteda.io/articles/quickstart.html).

Tab. 9.8 Keywords-in-context-Analyse für das Wort „Daten“

Doc	Pretext	Keyword	Posttext
RF1.txt	häufigsten zu Verwendung der weitergegebenen	Daten	gemacht wird ist, dass
RF1.txt	wird ist, dass die	Daten	zur sogenannten Benutzeroptimierung genutzt werden
RF1.txt	geschaltet wird. Zu diesen	Daten	sollten, meiner Meinung nach
RF1.txt	werden. Sollten jedoch weitere	Daten	, wie zum Beispiel Adresse
RF1.txt	darüber haben, was für	Daten	er angeben möchte und wie
RF1.txt	den AGB's angegeben ist welche	Daten	wofür genutzt werden, sollte
RF1.txt	ich an der Weitergabe meiner	Daten	sehe ist, dass man
RF1.txt	Jedoch wird jede Weitergabe von	Daten	auch dritten Personen einen Zugang

Quelle: Eigene Darstellung

Die bislang besprochenen Verfahren zeichnen sich allesamt dadurch aus, dass sie zwar harte Zahlen über die Verteilung von Wörtern oder Kombinationen in einem Korpus liefern, dabei aber immer explorativ und interpretationsbedürftig sind. Die Berechnung von Wortfrequenzen, Wahrscheinlichkeiten, Tf-idf-Werten, Kookkurenzen oder Pointwise Mutual Information unterstützt vorrangig die inhaltliche Auseinandersetzung mit einem Korpus, um einen Überblick über typische und seltene Fälle zu erhalten, sowie die Datenaufbereitung.

9.2 Diktionärbasierte Inhaltsanalyse

Führt man sich die Entstehung von Texten vor Augen, dann stellt ein Text immer nur eine Realisation von vielen äquivalenten Möglichkeiten dar. Die Abneigung oder Zustimmung zu einem Sachverhalt kann beispielsweise auf ganz unterschiedliche Weise ausgedrückt werden. Menschen haben einen umfangreichen Wortschatz lobender oder beleidigender Wörter und können sich mal einfach und mal elaboriert ausdrücken. Themen wie Datenschutz, die Entstehung der Menschheit oder Wahlen lassen sich auf sehr unterschiedliche Weise besprechen. Während die bislang dargestellte Herangehensweise – Auszählen von Wörtern – bei den konkreten Texten startet und auf Bedeutungen schließt, geht eine diktionärsbasierte Inhaltsanalyse umgekehrt vor. Hier stehen am Anfang der Analyse sogenannte latente Konstrukte wie Einstellungen, Themen oder auch spezielle Konstrukte wie

Tab. 9.9 Beispiele für negative und positive Wörter aus dem SentiWS

lemma	pos	sentiment	inflections
Gefahr	NN	1,00	Gefahren
Schuld	NN	−0,97	Schulden
unnötig	ADJX	−0,95	unnötigstes,unnötigere,unnötige […]
schädlich	ADJX	−0,93	schädlicher,schädlicheren […]
schwach	ADJX	−0,92	schwächstem,schwächsten […]
schämen	VVINF	−0,89	schämt,geschämt,schämtest […]
gelungen	ADJX	1,00	gelungnerem,gelungenster […]
perfekt	ADJX	0,73	perfekterer,perfekteren,perfektes […]
Lob	NN	0,72	Loben,Lobs,Lobes,Lobe
wunderbar	ADJX	0,72	wunderbarerem,wunderbarste […]
spannend	ADJX	0,72	spannendste,spannender […]
wunderschön	ADJX	0,70	wunderschönen,wunderschönsten […]

Quelle: Eigene Darstellung basierend auf dem SentiWS (Deutscher Wortschatz 2022)

Inzivilität und Diskursqualität, die dann im zweiten Schritt so operationalisiert werden, dass manifeste sprachliche Äußerungen als Indikator für diese Konstrukte verwendet werden. Latente Konstrukte umfassen also die nicht direkt messbaren Prozesse und Zustände, von denen man annimmt, dass sie hinter den beobachtbaren, manifesten sprachlichen Zeichen stehen. Der Vorgang, die latenten Konstrukte messbar zu machen, wird Operationalisierung genannt (Döring und Bortz 2016, S. 224).

Eine einfache Möglichkeit der Operationalisierung besteht darin, eine Reihe von Wörtern festzulegen, die beispielsweise für eine positive oder eine negative Einstellung stehen. Diese Form der Analyse, bei der die Valenz von Texten bestimmt wird, nennt sich Sentimentanalyse. Hierfür stehen einige wenige in wissenschaftlichen Projekten bereits erprobte Wörterbücher zur Verfügung. Ein deutschsprachiges Wörterbuch ist der SentimentWortschatz (Goldhahn et al. 2012), welcher von der Projektseite der Universität Leipzig heruntergeladen werden kann.[14] Die Daten gehen auf eines der ersten Digital-Humanities-Projekte zur automatisierten Textanalyse zurück. Das Diktionär des Programms General Inquirer (Stone et al. 1966) wurde mit Wörtern aus Zeitungskorpora unter Zuhilfenahme von Verfahren wie Kookkurrenzanalyse ergänzt und schließlich durch menschliche Kodierer:innen überprüft. In diesem Diktionär sind zu jeweils einem Lemma die Wortart (= PoS; Part-of-Speech), ein negativer bis positiver Sentimentwert und flektierte Wortformen enthalten (Tab. 9.9). Verbindet man diese Wortliste mit den Wörtern in einem tokenisierten Text, lässt sich

[14] Siehe Deutscher Wortschatz (2022; https://wortschatz.uni-leipzig.de/).

die Anzahl positiver oder negativer Wörter in einem Text als Indikator für die Valenz des Textes verstehen. Dass diese einfache Grundidee eine Reihe von Komplexitäten birgt, wird das folgende R-Beispiel verdeutlichen.[15]

Schritt 1: Texte einlesen Lesen Sie zunächst wie oben beschrieben das Korpus ein, tokenisieren Sie die Texte und führen Sie ein Stemming durch:

```
texte <- readtext("korpus", encoding = "UTF-8")

woerter <- unnest_tokens(
    texte, wort, text, to_lower = T
  ) %>%
  mutate(stem = wordStem(wort, language = "de"))
```

Im Ergebnis liegt ein Dataframe `woerter` vor, in der ersten Spalte wird für jeden Text eine `doc_id` vermerkt und in der letzten Spalte `stem` sind fortlaufend die Wortstämme enthalten.

Schritt 2: Wörterbuch einlesen Das Diktionär (☛ *Repositorium*) muss für den Abgleich auf die gleiche Weise wie das Textkorpus vorbereitet werden, das heißt, die Lemmata werden in Kleinschreibung umgewandelt und auf die Wortstämme zurückgeführt:

```
sentiws <- read_csv("sentiws/SentiWS20.csv", na = "")

sentiws <- sentiws %>%
    mutate(wort = str_to_lower(lemma)) %>%
    mutate(stem = wordStem(wort, language = "de"))
```

Dadurch entsteht ein Dataframe, der unter anderem eine Spalte `stem` und zu jedem Wortstamm in der Spalte `sentiment` einen Sentimentwert zwischen −1 und +1 enthält. Durch die Aufbereitung ist es möglich, dass ein Wortstamm doppelt, aber mit unterschiedlichen Sentiments, auftritt. Damit eine eindeutige Zuordnung zwischen den Texten und dem Diktionär möglich ist, können Sie die Wortstämme so zusammenfassen, dass nur das mittlere Sentiment verzeichnet ist. Das leistet die Funktion `summarise()` in Kombination mit `mean()`, wobei die Zu-

[15] Diktionärsbasierte Verfahren müssen immer validiert werden, zur Validität von Sentimentanalysen siehe van Atteveldt et al. (2021).

sammenfassung mit `group_by()` für jeden Wortstamm getrennt durchgeführt wird (dieses Muster wird in Abschn. 5.1 erklärt):

```
sentiws_mean <- sentiws %>%
  group_by(stem) %>%
  summarise(sentiment = mean(sentiment)) %>%
  ungroup()
```

Zur weiteren Vereinfachung der Analyse wird noch eine neue Spalte `sent_pos` eingeführt, in der mit den Wahrheitswerten `TRUE` und `FALSE` (kurz: `T` und `F`) vermerkt wird, ob es sich um ein Wort mit einem positiven oder negativen Sentimentwert handelt:

```
sentiws_mean <- sentiws_mean %>%
  mutate(sent_pos = ifelse(sentiment > 0, T, F)) %>%
  mutate(sent_neg = ifelse(sentiment < 0, T, F))
```

Im nächsten Schritt kann ausgezählt werden, wie viele Wörter in einem Text positiv oder negativ konnotiert sind. Diese Vorbereitungsschritte für das Korpus und das Diktionär zeigen bereits auf, dass eine Sentimentanalyse häufig stark vereinfachend vorgeht und entsprechend vorsichtig interpretiert werden muss. Die Ambivalenz von Wörtern (wenn sie gleichzeitig positive und negative Aspekte beinhalten) und auch Synonyme (wenn ein Wort verschiedene Bedeutungen hat) werden geglättet. Eine solche automatisierte Textanalyse ist zwar objektiv in dem Sinne, dass sie sich unabhängig von den ausführenden Wissenschaftler:innen wiederholen lässt. Trotzdem ergibt sich die Bedeutung von Wörtern, Sätzen und Texten aus dem Wechselspiel von Mitteilung und Rezeption und kann deshalb nicht nur von Rezipient:in zu Rezipient:in unterschiedlich ausfallen, sondern auch ein und die gleiche Person kann morgens zu einer anderen Einschätzung kommen als am Abend.

Schritt 3: Wörter den Texten zuordnen Sind Korpus und Wörterbuch auf die gleiche Art und Weise aufbereitet, können die beiden Tabellen verbunden werden. Jedem Wort aus den Texten lässt sich über einen `left_join()` ein Eintrag aus dem Wörterbuch zuordnen, indem die Spalte `stem` abgeglichen wird (zu Joins siehe Abschn. 4.2.2):

```
woerter_sentiment <- woerter %>%
  left_join(sentiws_mean, by = "stem")
```

Tab. 9.10 Zuordnung von Wörtern zu Sentiments (Auszug)

doc_id	wort	stem	sentiment	sentpos
RF1.txt	verletzt	verletzt	−0,5202	FALSE
RF1.txt	hackerangriffes	hackerangriff		
RF12.txt	abzulehnen	abzulehn		
RF1.txt	schon	schon	0,0081	TRUE
RF30.txt	schön	schon	0,0081	TRUE
RF1.txt	größeren	gross	0,1867	TRUE
RF1.txt	firmen	firm		

Quelle: Eigene Darstellung basierend auf dem SentiWS (Deutscher Wortschatz 2022)

Das Ergebnis besteht aus einem Dataframe, in den ersten Spalten finden sich die Daten aus der Wörterliste in `doc_id` und `wort`, in den letzten Spalten die Angaben `sentiment` und `sentpos` aus dem Diktionär, dazu die Spalte `stem`, über die der Abgleich vorgenommen wurde. Betrachtet man einen Ausschnitt aus dem Ergebnis (Tab. 9.10), fällt auf, dass nicht alle Wörter im Diktionär verzeichnet sind – im verwendeten Beispielkorpus lassen sich 17 % der Wörter zuordnen. Darunter sind viele Wörter wie „Firmen", denen sich nicht ohne Weiteres eine Tendenz zuordnen lässt, aber auch Wörter wie „Hackerangriff", die zwar intuitiv positiv bzw. negativ bewertet werden können, die aber für das Diktionär zu speziell sind. Während einige Zuordnungen wie ein negatives Sentiment zum Wort „verletzt" durchaus gut nachvollziehbar sind, treten auch Unschärfen auf. Das Wort „schon" wird durch die Zurückführung auf einen Wortstamm mit dem gleichen Sentiment wie das Wort „schön" belegt. Und das als positiv erkannte Wort „größeren" ließe sich je nach Kontext auch negativ interpretieren, etwa wenn es um einen größeren Schaden geht.

Schritt 4: Auswertung Unschärfen bedeuten nicht zwangsläufig, dass keine sinnvollen Aussagen getroffen werden können. Wichtig ist vor allem, dass jedes Wort lediglich als Indikator verstanden wird – wenn viele Indikatoren gleichzeitig betrachtet werden, gleichen sich die Fehlzuordnungen in die positive oder negative Richtung im besten Fall aus. Deshalb werden die Sentiments im nächsten Schritt so zusammengefasst, dass für jeden Text die Anzahl aller Wörter `n_token`, die Anzahl positiv erkannter Wörter `n_pos` und die Anzahl negativ erkannter Wörter `n_neg` ausgezählt wird. Im folgenden Beispiel wird dazu ein Trick angewendet: Der Wahrheitswert `TRUE` wird bei der Addition automatisch in `1` umgewandelt, sodass sich als Summe der Wahrheitswerte die Anzahl der in den Spalten `sent_pos` bzw. `sent_neg` mit `TRUE` gekennzeichneten Fälle ergibt. Mit `na.rm=T` werden die fehlenden Werte ignoriert. Daraus lässt sich für jedes Dokument der

Anteil positiv erkannter Wörter `p_pos` und negativ erkannter Wörter `p_neg` berechnen:

```
texte_sentiment <- woerter_sentiment %>%
  group_by(doc_id) %>%

  summarize(
      n_token = n(),

      n_pos = sum(sent_pos, na.rm = T),
      n_neg = sum(sent_neg, na.rm = T),

      p_pos = n_pos / n_token,
      p_neg = n_neg / n_token
  ) %>%
  ungroup()
```

Schlussendlich entsteht ein Dataframe, in dem für jeden Text der Umfang positiver und negativer Wörter verzeichnet ist, was sich gut visualisieren lässt. Trägt man den Anteil positiver Wörter auf der X-Achse und die negativen Wörter entsprechend auf der Y-Achse ab, kann jedes Dokument als Punkt eingetragen werden:

```
texte_sentiment %>%
  ggplot(aes(x = p_pos, y = p_neg)) +
  geom_point()
```

Es zeigt sich dabei wieder, dass die Welt nicht schwarz und weiß ist (Abb. 9.2). Texte können eindeutig negativ (links oben) oder positiv (rechts unten) eingeordnet werden, aber eine Vielzahl an Dokumenten bleibt ambivalent (rechts oben) oder indifferent (links unten), weil gleichzeitig positive und negative Bewertungen vorgenommen werden bzw. nur wenige bewertende Wörter enthalten sind.

Der Mittelwert positiver Sentiments liegt mit 0,13 deutlich über dem Wert von 0,04 für das mittlere negative Sentiment (in der Grafik durch Linien gekennzeichnet). Ob das tatsächlich darauf hindeutet, dass die meisten der befragten Studierenden – das Korpus umfasst Reflexionstexte zum Thema Datenschutz – eine positive Einstellung ausdrücken, ist allerdings fraglich. Bei der Interpretation sollte besser vergleichend vorgegangen werden. Im Vergleich zu anderen Texten lassen sich über- oder unterdurchschnittlich positive oder negative Texte identifizieren. Auf diese Weise können nicht nur tendenziell positivere oder negativere, sondern eben auch unentschiedene oder mehrdeutige Texte für eine weitere Analyse identifiziert werden.

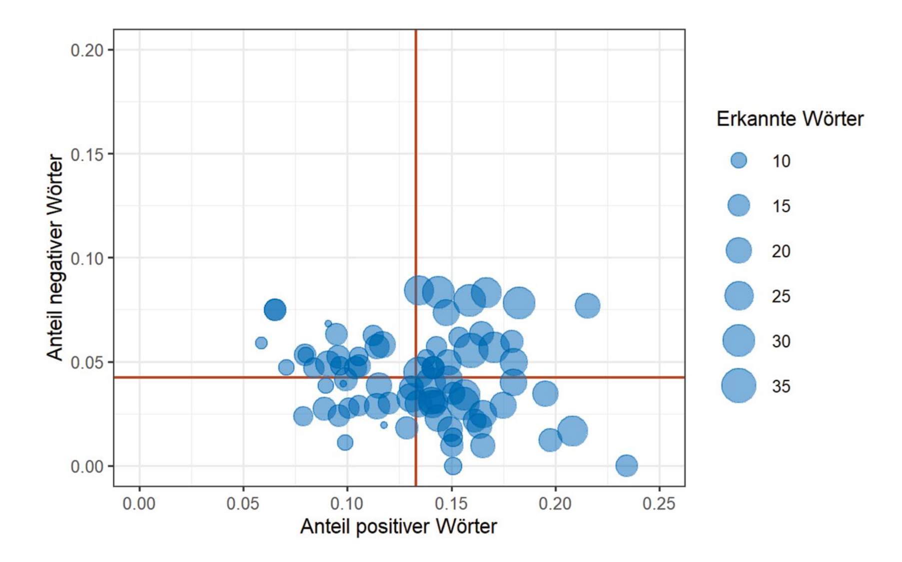


Abb. 9.2 Sentiments eines Beispielkorpus. (Quelle: eigene Darstellung)

Das am Beispiel einer Sentimentanalyse verdeutlichte Prinzip lässt sich mit entsprechenden Wörterbüchern auf andere Anwendungsfälle übertragen. Häufig verwendet wird beispielsweise das LIWC-Diktionär (Linguistic Inquiry and Word Count), in dem psychologische Kategorien wie etwa Emotionen operationalisiert sind.[16] Die deutsche Version des LIWC ist für wissenschaftliche Analysen bei den Autoren erhältlich (Wolf et al. 2008). Auch für spezielle und aktuelle Fälle wie die Erkennung von Hate Speech werden fortlaufend Diktionäre entwickelt,[17] wobei entsprechende Daten häufig zunächst für die englische Sprache vorliegen und damit nicht ohne Übersetzung für deutschsprachige Analysen eingesetzt werden können. Es lassen sich aber auch eigene Wörterbücher entwickeln – um aus größeren Textkorpora relevante Texte auszuwählen, reichen mitunter einfache Stichwortlisten.

[16] LIWC ist ein Programm zur Textanalyse, das nicht kostenlos zur Verfügung steht, siehe Pennebaker et al. (2022; https://www.liwc.app/). Das darin verwendete LIWC-Diktionär kann aber auf Nachfrage bei einem der Autoren (James W. Pennebaker) für akademische Zwecke kostenlos bezogen werden.

[17] Für eine Übersicht siehe Vidgen und Derczynski (2020). Siehe auch Derczynski et al. (2022; https://github.com/leondz/hatespeechdata).

9.3 Syntax und Semantik

In den vorangegangenen Abschnitten wurden als Analyseeinheiten immer nur einzelne Wörter und Wortkombinationen betrachtet. Sprache besteht allerdings nicht aus isolierten Wörtern, die Wörter stehen zueinander in Beziehung und diese Beziehungen vermitteln Informationen, die für ein Verständnis von Texten nötig sind. Löst man sich bei der automatisierten Textanalyse von einzelnen Wörtern und betrachtet diese im Kontext von anderen Wörtern, kommen Syntax und Semantik in Spiel. Syntax beschreibt die Regeln, nach denen Wörter und Sätze zusammengebaut werden, während sich die Semantik damit beschäftigt, was diese Wörter und Sätze bedeuten (Bußmann 1990, S. 672, 766). Unter dem Stichwort *Natural Language Processing* (NLP) werden Verfahren der automatisierten Textanalyse zusammengefasst, die syntaktische und semantische Strukturen berücksichtigen. Zwar stoßen automatisierte Verfahren zum Verarbeiten menschlicher Sprache immer auch an Grenzen, die im Folgenden angesprochenen Methoden gehören aber mittlerweile zum Standardrepertoire der Computational Methods.

9.3.1 Natural Language Processing

Ein erster Schritt, um über den Bag-of-Words-Ansatz hinauszugehen, besteht darin, die Wortarten zu berücksichtigen. Beim sogenannten **Part-of-Speech-Tagging (POS-Tagging)** werden beispielsweise Artikel und Substantive unterschieden (einführend zum Beispiel Martinez 2012). Diese Information kann hilfreich sein, um nur bedeutungstragende Wortarten (Substantive, Verben, Adjektive) in eine Analyse einzuschließen, anstatt eine Liste von Stoppwörtern auszuschließen. Das Bestimmen der Wortarten ist allerdings nicht trivial, wie man sich an der Unterscheidung von Substantiven und Verben vor Augen führen kann. Auch wenn es etwa naheliegen mag, Substantive und Verben zum Beispiel anhand der Groß- und Kleinschreibung zu unterscheiden, reicht dieses Unterscheidungskriterium allein nicht aus. Verben werden im Deutschen am Satzanfang großgeschrieben, in der konzeptionell mündlichen Sprache von Online-Chats finden sich kleingeschriebene Substantive und in Sprachen wie dem Englischen kann ein kleingeschriebenes „walk" sowohl den Spaziergang als auch spazierengehen bezeichnen. Deshalb werden im besten Fall automatisierte Klassifikationsverfahren eingesetzt, die über einfache Regelformulierungen hinausgehen (siehe Abschn. 8.2). Ein Spezialfall des POS-Taggings ist das Erkennen von Eigennamen durch **Named-Entity-Recognition (NER)**. Dieses

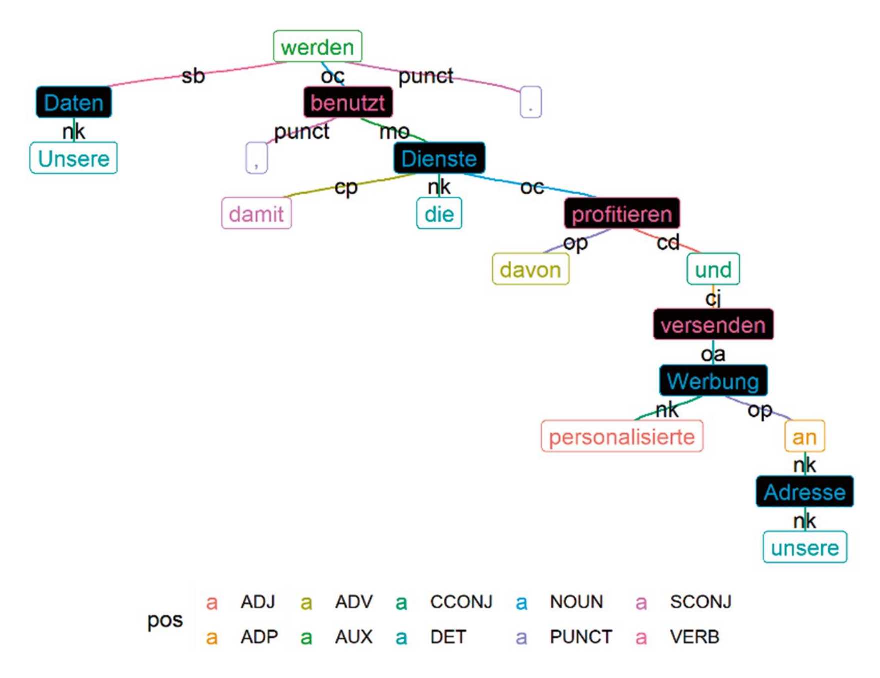


Abb. 9.3 Dependenzparsing. Der Satz wurde mit spacyR geparsed und mit ggraph visualisiert. (Quelle: eigene Darstellung)

automatische Klassifikationsverfahren dient der Identifizierung von Personen, Orten oder Organisationen in Texten.

Wortarten ergeben sich teilweise aus den Beziehungen zwischen Wörtern. Im Deutschen stehen etwa Artikel immer vor den Substantiven, sie treten in der Regel nicht unabhängig voneinander auf. Führt man den Gedanken weiter, so lässt sich die gesamte syntaktische Struktur eines Satzes rekonstruieren. **Dependenzparser** (einführend zum Beispiel Nederhof und Satta 2010) bilden die Satzstruktur so in einem Baum ab, dass die Abhängigkeiten zwischen den Wörtern sichtbar werden (Abb. 9.3). Betrachtet man die Verbindungen von Adjektiven und Substantiven, so lassen sich Prädikationen wie „personalisierte Werbung“ finden. Im Deutschen bilden Verben den Mittelpunkt von Aussagen, Verbalkonstruktionen können zur Rekonstruktion von Handlungen und Ereignissen[18] herangezogen werden, wie sie in

[18] Zur Analyse von Handlungen siehe zum Beispiel van Atteveldt et al. (2017). Ereignisse werden beispielsweise im GDELT-Projekt automatisiert aus der Berichterstattung extrahiert, siehe Leetaru (2021; https://www.gdeltproject.org/).

den Teilsätzen „unsere Daten werden benutzt“ oder „die Dienste profitieren davon“ formuliert sind. Eine Alternative zum Dependenzparsing besteht darin, einen Satz so in seine Bestandteile zu zerlegen, dass einzelne voneinander abhängende Phrasen identifizierbar werden. Diese sogenannten **Konstituenten** (*constituency parsing*) helfen dabei, Nebensätze oder indirekte Rede zu identifizieren.

Wörter stehen nicht nur in syntaktischer, sondern auch in semantischer Relation zueinander. Das wird etwa bei Synonymen sichtbar: „Apfelsine“ und „Orange“ bezeichnen den gleichen Gegenstand, sie werden mit einem Bag-of-Words-Ansatz aber als unterschiedliche Wörter behandelt. In der automatisierten Sprachanalyse können zur Lösung dieses Problems **Word Embeddings** eingesetzt werden. Jedes Wort wird dabei durch einen Vektor von Eigenschaften repräsentiert (Turney und Pantel 2010). Beispielsweise sind Apfelsinen und Orangen beide essbar, Steine dagegen nicht. Diese Eigenschaft kann man indirekt aus dem sprachlichen Kontext erschließen. So sind die Kombinationen „Orangen essen“ oder „Apfelsinen essen“ erwartbar und unterscheiden sich von einem sprachlichen Kontext wie „Steine werfen“. Die Bedeutung eines Wortes ergibt sich also aus dessen Verwendung – diese Erkenntnis wird als Distributionshypothese bezeichnet: „You shall know a word by the company it keeps“ (Firth 1962, S. 11).

Analysiert man die Verwendung von Wörtern in großen Textkorpora, so lassen sich daraus Dimensionen gewinnen, auf denen sich Wörter verorten lassen – die sogenannten Word Embeddings. Für jedes Wort wird ein Vektor mit Eigenschaften gebildet. In der Textanalyse kann dann mit diesen in Zahlen ausgedrückten Dimensionen anstelle der Wörter weitergearbeitet werden. Werden Wörter durch Vektoren repräsentiert, lassen sich auch Ähnlichkeiten bestimmen. So sollten sich die Vektoren von „Apfel“ und „Orange“ ähnlicher sein als die Vektoren von „Apfel“ und „Stein“. Multilinguale Word Embeddings, bei denen zum Beispiel „book“ und „Buch“ durch ähnliche Vektoren erfasst sind, ermöglichen zudem sprachübergreifende Analysen. Auch Gegensätze lassen sich ermitteln, wenn zwei Wörter in vielen Dimensionen ähnlich, in einigen Dimensionen aber gegenteilig sind. Ein klassisches Beispiel dafür ist die Beziehung zwischen den Wörtern „König“ und „Königin“, die sich lediglich im Geschlecht unterscheiden (Pennington et al. 2014, S. 1532). Um weitere Beziehungen zwischen Begriffen, insbesondere über- und untergeordnete Begriffe (Hyperonyme und Hyponyme), zu berücksichtigen, können zudem strukturierte Ontologien eingesetzt werden, wie sie etwa von WordNet[19] entwickelt wurden (siehe auch Abschn. 3.7).

[19] Siehe Princeton University (2010; https://wordnet.princeton.edu/).

9.3.2 Textanalyse mit spaCy

Die meisten der genannten Techniken lassen sich vergleichsweise unkompliziert mit etablierten Packages für R oder Python umsetzen. Diese Packages parsen einen Text, das heißt, sie legen ihn in einem Datenformat ab, das die ursprüngliche Struktur des Textes abbildet und ggf. um zusätzliche Informationen wie Part-of-Speech-Tags ergänzt. Zahlreiche Funktionalitäten stellt beispielsweise die Open-Source-Software *spaCy* bereit.[20] SpaCy ist für Python entwickelt worden, lässt sich aber dank des *spacyR*-Package (Benoit und Matsuo 2020) gut in R nutzen.[21]

Benötigt wird für *spacyR* eine Python-Distribution – installieren Sie sich zunächst die Python-Distribution Miniconda. Wenn Sie bereits eine Anaconda-Distribution (siehe Abschn. 5.2) auf Ihrem Computer eingerichtet haben, können Sie diesen Schritt überspringen. Anschließend installieren und laden Sie in R das Package *spacyr*. Über die Funktion `spacy_install()` wird automatisch eine virtuelle Conda-Umgebung mit Python und spaCy eingerichtet:[22]

```
library(spacyr)
spacy_install()
```

Beim Parsen greift SpaCy auf vortrainierte Modelle zurück (siehe Abschn. 8.1 zu den Grundlagen automatisierter Klassifikation). Ein deutsches Sprachmodell können Sie mit folgendem Befehl herunterladen:

```
spacy_download_langmodel("de")
```

[20] Siehe Honnibal und Montani (2020; https://spacy.io/). Die Ergebnisse können dann mit weiteren Packages verarbeitet werden, etwa zur Extraktion von Phrasen mit *rsyntax* (van Atteveldt und Welbers 2022).

[21] In solchen Situationen, in denen Funktionen in einer Programmiersprache geschrieben, aber aufgrund ihrer Prominenz auch für andere Sprachen verfügbar gemacht werden sollen, kommen sogenannte Wrapper zum Einsatz. Wrapper schlagen also eine Brücke von einer Programmiersprache in die andere.

[22] Miniconda (Anaconda 2022; https://docs.conda.io/en/latest/miniconda.html) ist eine Minimalversion der Python-Distribution Anaconda, die im Kern vor allem den Paketmanager *conda* und Python mitbringt. Siehe Benoit und Matsuo (2020; https://cran.r-project.org/web/packages/spacyr/readme/README.html) für weitere Informationen zur Installation von Spacy für R.

Tab. 9.11 Ergebnis des Parsings mit SpaCy

doc_id	sentence_id	token_id	token	lemma	pos	head_token_id	dep_rel
RF29	9	1	Unsere	mein	DET	2	nk
RF29	9	2	Daten	Datum	NOUN	3	sb
RF29	9	3	werden	werden	AUX	3	ROOT
RF29	9	4	benutzt	benutzen	VERB	3	oc
RF29	9	5	,	,	PUNCT	4	punct
RF29	9	6	damit	damit	SCONJ	10	cp
RF29	9	7	die	der	DET	8	nk
RF29	9	8	Dienste	Dienst	NOUN	10	sb
RF29	9	9	davon	davon	ADV	10	op
RF29	9	10	profitieren	profitieren	VERB	4	mo

Quelle: Eigene Darstellung

Sind alle Komponenten heruntergeladen und installiert – diese Schritte müssen nur einmalig durchlaufen werden –, kann der Parser mit dem deutschen Sprachmodell initialisiert werden. SpaCy ist dann eingerichtet und einsatzbereit:

```
spacy_initialize(model = "de_core_news_sm")
```

Wenn Sie Texte mit der `readtext()`-Funktion einlesen, können diese anschließend an die Funktion `spacy_parse()` übergeben werden, um Wortarten oder Dependenzstrukturen zu bestimmen:

```
library(tidyverse)
library(readtext)

texte <- readtext("korpus", encoding = "UTF-8")
txt_parsed <- spacy_parse(texte, dependency = TRUE)
```

Der Output enthält im Dataframe `txt_parsed` die tokenisierten und annotierten Texte (Tab. 9.11). Jedes einzelne Token innerhalb eines Satzes wurde in der Spalte `token_id` durchnummeriert, lemmatisiert und in der Spalte `pos` mit einer Wortart versehen. Die verwendeten Bezeichnungen stammen aus dem TIGER-Annotationsschema.[23] Die syntaktische Struktur der Sätze ist in den Spalten `head_token_id` und `dep_rel` erfasst. Erstere enthält die ID des syntaktisch übergeordneten Tokens, letztere eine Klassifikation der Beziehung. Im Bei-

[23] Auf der Seite des TIGER-Projekts findet sich neben der Dokumentation auch ein Teil des Trainingsmaterials, das in das Sprachmodell eingeflossen ist, siehe IMS (2003; https://www.ims.uni-stuttgart.de/forschung/ressourcen/korpora/tiger/).

spiel wird etwa das Hilfsverb „werden“ (`token_id` = 3) als Wurzelelement (root) des Satzes identifiziert, davon hängt das Verb „benutzt“ (`head_token_id` = 3) ab. Das Wort „Daten“ wird in der Spalte `deep_rel` als Subjekt (sb) des Satzes gekennzeichnet.

Auch wenn ein linguistisches Grundwissen für die Arbeit mit diesen Daten hilfreich ist, lassen sich schnell einfache Analysen anschließen. Ermitteln Sie die häufigsten Wortarten, filtern Sie bedeutungstragende Wortarten heraus oder extrahieren Sie Phrasen rund um das Wort „Daten“!

Übungsfragen

1. Worin unterscheiden sich Textanalyse und Inhaltsanalyse?
2. Welche Vor- und Nachteile ergeben sich aus dem Bag-of-Words-Ansatz?
3. Suchen Sie sich zwei beliebige Sätze aus diesem Kapitel und bestimmen Sie die Anzahl der Types und der Token!
4. Welche Schritte werden bei der Aufbereitung von Texten typischerweise durchgeführt?
5. Was versteht man unter Tokenisierung?
6. Finden Sie heraus: Wie kann ein Text mit R in einzelne Sätze zerlegt werden?
7. Wie lassen sich PDF-Dateien mit R einlesen?
8. Was ist der Unterschied zwischen einem Unigram und einem Trigram?
9. Was sagt der Tf-idf-Wert aus und wofür kann er verwendet werden?
10. Wie kann die Kookkurrenz von Wörtern bestimmt werden?
11. Was versteht man unter POS-Tagging und unter NER?
12. Was ist ein Dependenzparser?

Weiterführende Literatur

Aggarwal, C.C. (2018). *Machine learning for text.* Cham: Springer.

Aggarwal, C.C., & Zhai, C. (Hrsg.) (2012). *Mining text data.* New York: Springer.

Lemke, M. & Wiedemann, G. (Hrsg.) (2016). *Text Mining in den Sozialwissenschaften: Grundlagen und Anwendungen zwischen qualitativer und quantitativer Diskursanalyse.* Wiesbaden: Springer.

Grimmer, J., Roberts, M. E. & Stewart, B. M. (2022). Text as data: A new framework for machine learning and the social sciences. Princeton: Princeton University Press.

Silge, J. & Robinson, D. (2017). *Text mining with R. A tidy approach.* Beijing: O'Reilly. http://tidytextmining.com

Wiedemann, G. (2016). *Text Mining for Qualitative Data Analysis in the Social Sciences. A Study on Democratic Discourse in Germany*. Wiesbaden: Springer.

Literatur

Anaconda. (2022). Miniconda (Version 3) [Computer software]. https://docs.conda.io/en/latest/miniconda.html

Anthony, L. (2022). AntConc. A freeware corpus analysis toolkit for concordancing and text analysis (Version 4.0.10) [Computer software]. https://www.laurenceanthony.net/software/antconc/

ATLAS.ti Scientific Software Development GmbH. (2022). ATLAS.ti (Version 22) [Computer software]. https://atlasti.com

Benoit, K. & Matsuo, A. (2020). spacyr. Wrapper to the 'spaCy' 'NLP' Library (Version 1.2.1) [Computer software]. https://cran.r-project.org/web/packages/spacyr/readme/README.html

Benoit, K., Muhr, D. & Watanabe, K. (2021). stopwords. Multilingual Stopword Lists (Version 2.3) [Computer software]. https://cran.r-project.org/package=stopwords

Benoit, K., Obeng, A., Watanabe, K., Matsuo, A., Nulty, P. & Müller, S. (2021). readtext. Import and Handling for Plain and Formatted Text Files (Version 0.81) [Computer software]. https://cran.r-project.org/package=readtext

Benoit, K., Watanabe, K., Wang, H., Nulty, P., Obeng, A., Müller, S. et al. (2022). *quanteda: An R package for the quantitative analysis of textual data. Quick Start Guide*. Zugriff am 08.05.2022. https://quanteda.io/articles/quickstart.html

Bouchet-Valat, M. (2020). SnowballC: Snowball Stemmers Based on the C 'libstemmer' UTF-8 Library (Version 0.7.0) [Computer software]. https://cran.r-project.org/package=SnowballC

Bouma, G. (2009). Normalized (pointwise) mutual information in collocation extraction. In C. Chiarcos, R. E. de Castilho & M. Stede (Hrsg.), *Proceedings of the Biennial GSCL Conference 2009. From form to meaning: processing texts automatically* (S. 43–53). Tübingen: Narr.

Bußmann, H. (1990). *Lexikon der Sprachwissenschaft*. Stuttgart: Kröner.

Derczynski, L., Vidgen, B., Kirk, H. R., Johansson, P., Chung, Y.-L., Guldborg, M., Kongsbak, K., Sprejer, L., & Zeinert, P. (2022). Hate Speech Dataset Catalogue. Zugriff am 05.01.2023. https://github.com/leondz/hatespeechdata

Deutscher Wortschatz. (2022). *Wortschatz-Portal*. Zugriff am 08.05.2022. https://wortschatz.uni-leipzig.de/de

Döring, N. & Bortz, J. (2016). *Forschungsmethoden und Evaluation in den Sozial- und Humanwissenschaften* (5. Aufl.). Berlin: Springer. https://doi.org/10.1007/978-3-642-41089-5

Dunning, T. (1993). Accurate Methods for the Statistics of Surprise and Coincidence. *Computational Linguistics*, *19*(1), 61–74.

Firth, J. R. (1962). A synopsis of linguistic theory 1930–1955. In J. R. Firth, W. Haas, M. Halliday, W. Allen & R. H. Robins (Hrsg.), *Studies in linguistic analysis* (S. 1–32). Oxford: Blackwell.

Goldhahn, D., Eckart, T. & Quasthoff, U. (2012). Building Large Monolingual Dictionaries at the Leipzig Corpora Collection: From 100 to 200 Languages. *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC'12)*, 759–765. http://lrec-conf.org/proceedings/lrec2012/pdf/327_Paper

Honnibal, M. & Montani, I. (2020). spaCy: Industrial-Strength Natural Language Processing in Python (Version 3.3) [Computer software]: Explosion. https://spacy.io/

IMS. Institut für Maschinelle Sprachverarbeitung, Universität Stuttgart. (2003). *TIGER Korpus 2.2*. Zugriff am 08.05.2022. https://www.ims.uni-stuttgart.de/forschung/ressourcen/korpora/tiger/

Jünger, J., Geise, S. & Hänelt, M. (2022). Unboxing Computational Social Media Research From a Datahermeneutical Perspective: How Do Scholars Address the Tension Between Automation and Interpretation? *International Journal of Communication*, (16), 1482–1505.

Krippendorff, K. (2013). *Content analysis. An introduction to its methodology* (3. Aufl.). Los Angeles: Sage.

Leetaru, K. H. (2021). *The GDELT Project*. Zugriff am 08.05.2022. https://www.gdeltproject.org/

Le Pennec, E. & Slowikowski, K. (2019). ggwordcloud. A Word Cloud Geom for 'ggplot2' (Version 0.5.0) [Computer software]. https://cran.r-project.org/package=ggwordcloud

Lexical Computing. (2022). *Sketch Engine*. https://www.sketchengine.eu/

Martinez, A. R. (2012). Part-of-speech tagging. *Wiley Interdisciplinary Reviews: Computational Statistics*, *4*(1), 107–113. https://doi.org/10.1002/wics.195

Merten, K. (1995). *Inhaltsanalyse. Einführung in Theorie, Methode und Praxis* (2., verbesserte Aufl.). Wiesbaden: Springer.

Moretti, F. (2013). *Distant reading* (4. Aufl.). London: Verso.

Nederhof, M. & Satta, G. (2010). Theory of Parsing. In A. Clark, C. Fox & S. Lappin (Hrsg.), *The Handbook of Computational Linguistics and Natural Language Processing* (S. 105–130). Oxford: Wiley-Blackwell. https://doi.org/10.1002/9781444324044.ch4

Newman, M. E. (2005). Power laws, Pareto distributions and Zipf's law. *Contemporary Physics*, *46*(5), 323–351. https://doi.org/10.1080/00107510500052444

Peirce, C. S. (1906). Prolegomena to an apology for pragmaticism. *Monist*, *16*(4), 492–546. https://doi.org/10.5840/monist190616436

Pennebaker, J. W., Booth, R. J., Boyd, R. L. & Francis, M. E. (2022). Linguistic Inquiry and Word Count: LIWC-22 (Version 5) [Computer software]: Pennebaker Conglomerates. https://www.liwc.app/

Pennington, J., Socher, R. & Manning, C. (2014). GloVe: Global Vectors for Word Representation. In A. Moschitti, B. Pang & W. Daelemans (Hrsg.), *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)* (S. 1532–1543). Stroudsburg: Association for Computational Linguistics (ACL). https://doi.org/10.3115/v1/D14-1162

Piantadosi, S. T. (2014). Zipf's word frequency law in natural language: a critical review and future directions. *Psychonomic Bulletin & Review*, *21*(5), 1112–1130. https://doi.org/10.3758/s13423-014-0585-6.

Princeton University. (2010). *WordNet. A Lexical Database for English*. Zugriff am 08.05.2022. https://wordnet.princeton.edu/

Robinson, D., Misra, K. & Silge, J. (2021). widyr. Widen, Process, then Re-Tidy Data (Version 0.1.4) [Computer software]. https://cran.r-project.org/package=widyr

Silge, J. & Robinson, D. (2016). tidytext: Text Mining and Analysis Using Tidy Data Principles in R. *The Journal of Open Source Software*, *1*(3), 37. https://doi.org/10.21105/joss.00037

Silge, J. & Robinson, D. (2017). *Text mining with R. A tidy approach*. Sebastopol: O'Reilly.

Stone, P. J., Dunphy, D., Smith, M. S. & Ogilvie, D. M. (1966). *The General Inquirer. A Computer Approach to Content Analysis*. Cambridge: MIT Press.

Syncro Soft. (2022). Oxygen [Computer software]. https://www.oxygenxml.com/

Turney, P. D. & Pantel, P. (2010). From Frequency to Meaning: Vector Space Models of Semantics. *Journal of Artificial Intelligence Research*, *37*, 141–188. https://doi.org/10.1613/jair.2934

Van Atteveldt, W. & Welbers, K. (2022). Rsyntax. R library to help dealing with syntactic structure [Computer software]. https://github.com/vanatteveldt/rsyntax

Van Atteveldt, W., van der Velden, M. A. C. G. & Boukes, M. (2021). The Validity of Sentiment Analysis: Comparing Manual Annotation, Crowd-Coding, Dictionary Approaches, and Machine Learning Algorithms. *Communication Methods and Measures*, *15*(2), 121–140. https://doi.org/10.1080/19312458.2020.1869198

Van Atteveldt, W., Sheafer, T., Shenhav, S. R. & Fogel-Dror, Y. (2017). Clause Analysis. Using Syntactic Information to Automatically Extract Source, Subject, and Predicate from Texts with an Application to the 2008–2009 Gaza War. *Political Analysis*, *25*(2), 207–222. https://doi.org/10.1017/pan.2016.12

VERBI Software. (2021). MAXQDA (Version 2022) [Computer software]. https://www.maxqda.com/

Vidgen, B. & Derczynski, L. (2020). Directions in abusive language training data, a systematic review: Garbage in, garbage out. *PloS One*, *15*(12), e0243300. https://doi.org/10.1371/journal.pone.0243300

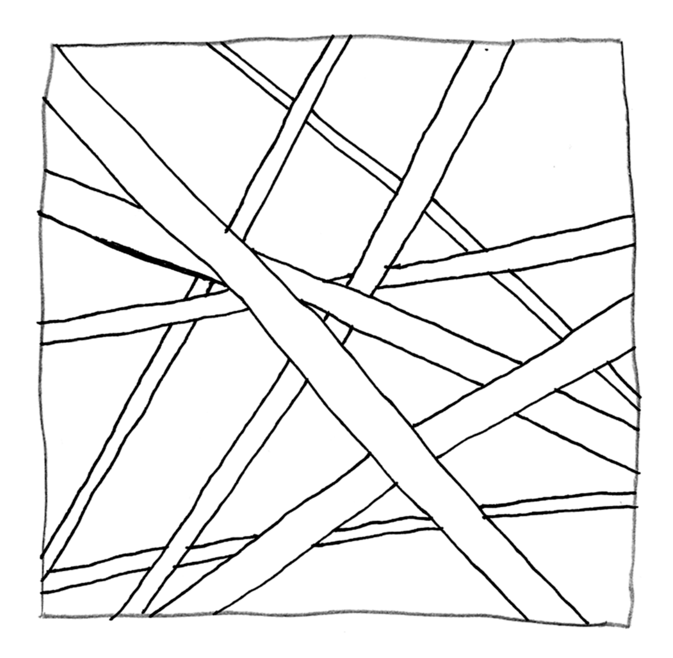
Wickham, H. (2019a). stringr. Simple, Consistent Wrappers for Common String Operations (Version 1.4.0) [Computer software]. https://cran.r-project.org/package=stringr

Wolf, M., Horn, A. B., Mehl, M. R., Haug, S., Pennebaker, J. W. & Kordy, H. (2008). Computergestützte quantitative Textanalyse. *Diagnostica*, *54*(2), 85–98. https://doi.org/10.1026/0012-1924.54.2.85



Netzwerkanalyse

10



J. Jünger, C. Gärtner, *Computational Methods für die Sozial- und Geisteswissenschaften*, https://doi.org/10.1007/978-3-658-37747-2_10

Zusammenfassung

Dieses Kapitel führt in die Methode der Netzwerkanalyse ein. Sie lernen Grundbegriffe und Kennwerte der Netzwerkanalyse kennen und erlernen Techniken zur Erhebung, Aufbereitung und Analyse von Netzwerkdaten.

Im Online-Repositorium unter https://github.com/strohne/cm finden Sie begleitend zum Kapitel weitere Materialien, auf die wir im Text mit ☛ verweisen.

Schlüsselwörter

Soziale Netzwerke · Semantische Netzwerke · Graphen · Knoten und Kanten · Komponenten · Dichte · Zentralität · Egozentrierte Netzwerke · Schneeballsampling · Empfehlungsnetzwerk · Facepager · igraph · Tidygraph · Gephi · Force-directed Layout

Die Welt lässt sich in vielen Bereichen als Netzwerk begreifen. Menschen stehen über Kommunikation in Beziehung zueinander, Begriffe stehen zueinander in semantischen Beziehungen und selbst die Abfolge von Ereignissen lässt sich als zeitliche Beziehung interpretieren (Albrecht 2013; Wasserman und Faust 1994, S. 9). Die Netzwerkanalyse bietet Werkzeuge und Ansätze, um solche Beziehungsdaten auszuwerten.

Netzwerkanalysen sind ein typisches Anwendungsfeld automatisierter Methoden und werden sowohl in den Sozial- als auch in den Geisteswissenschaften vielfältig eingesetzt (Amaral 2017; Cioffi-Revilla 2010, S. 260). Das liegt möglicherweise daran, dass mittlerweile in vielen Lebensbereichen umfangreiche Beziehungsdaten anfallen, die manuell kaum zu bewältigen sind. Es gibt aber noch einen weiteren Grund dafür, dass gängige statistische Verfahren hier an Grenzen stoßen. Klassischerweise wird in der Statistik meist unterstellt, dass Beobachtungen voneinander unabhängig sind. Das ist bei Beziehungsdaten grundsätzlich nicht der Fall – ganz im Gegenteil, die Rolle einer Person als Mutter ergibt sich erst daraus, dass sie auch mindestens ein Kind geboren, adoptiert oder umsorgt hat. Die Netzwerkanalyse stellt Methoden bereit, um solche Abhängigkeiten zu berücksichtigen.

Netzwerkanalyse ist dabei sowohl Methode als auch Theorie (Beckert 2005, S. 287 ff.). Sie umfasst unterschiedliche Verfahren, um Netzwerke zu konzipieren, zu beschreiben und auszuwerten. Gleichzeitig gehen mit methodischen Aspekten auch grundlegende theoretische Positionen einher, wie aus der Akteur-Network-

Theory (Latour 1996), der relationalen Soziologie (White 2008) oder aus Feldtheorien (Bourdieu 1985). Zum Beispiel führte die Analyse von Netzwerkdaten zu der theoretischen Erkenntnis der *Strength of Weak Ties*, wobei lose, schwache Bekanntschaftsbeziehungen im Gegensatz zu engen, starken Freundschaftsbeziehungen in sozialen Netzwerken eine entscheidende Relevanz für den Gesamtzusammenhalt des Netzwerks haben (Granovetter 1973).

In den Sozial- und Geisteswissenschaften können grundsätzlich drei Arten von Netzwerken unterschieden werden, die je andere Gegenstände als Netzwerk betrachten und entsprechend unterschiedliche Fragen aufwerfen:

- In **sozialen Netzwerken** interessieren Beziehungen zwischen **Akteuren** (Wasserman und Faust 1994, S. 20), also zwischen Einzelpersonen, kollektiven oder korporativen Akteuren. Dadurch kann beispielsweise betrachtet werden, wie sich soziale Ungleichheit formiert (Jansen 2003, S. 237 ff.) oder wie Identität in sozialen Bewegungen ausgehandelt wird (Diani und McAdam 2003).
- **Semantische Netzwerke** bilden stattdessen die Beziehungen zwischen **Konzepten** ab. So können beispielsweise Informationen zur Repräsentation von Wissen (Quillian 1967) oder Frames in Nachrichtenartikeln (Schultz et al. 2012) netzwerkanalytisch konzipiert und analysiert werden. Zur Modellierung von Wissensstrukturen eignet sich auch das Resource Description Framework (siehe Abschn. 3.7).
- Um Prozesse zu untersuchen, können diese als **raumzeitliche Netzwerke** modelliert werden. Hiermit kann unter anderem die Abfolge von Kommunikationsereignissen (Albrecht 2013; Malsch und Schlieder 2004) erfasst werden. Will man etwa Verläufe der Webseitennutzung analysieren, so lassen sich die Übergangswahrscheinlichkeiten von einer Webseite zu einer anderen als sogenannte Markov-Kette (Markov 2006) erfassen, um dann typische Verläufe zu extrahieren. Auch bei der Navigation greift man auf netzwerkanalytische Verfahren zurück, so lässt sich etwa der kürzeste Weg zwischen zwei Orten mit dem Dijkstra-Algorithmus (Dijkstra 1959) berechnen.[1]

[1] Tatsächlich hat Leonhard Euler bereits Anfang des 18. Jahrhunderts einen wichtigen Grundstein der Netzwerkanalyse gelegt, indem er ein Raumproblem graphentheoretisch löste: Das Königsberger Brückenproblem bestand in der Frage, ob es in Königstein einen Rundweg gibt, bei dem man jede von sieben Brücken genau einmal überquert (Euler 2000).

Im Kontext von Kommunikationsprozessen fallen alle drei Beziehungsarten immer zusammen. Kommunikation findet zwischen Akteuren statt, enthält Aussagen mit Referenzen auf Konzepte und ist durch einen Mitteilungs- und einen Rezeptionsakt raumzeitlich verortet. Dennoch kann es sinnvoll sein, diese Ebenen analytisch zu trennen und je nach Fragestellung einen Aspekt zu fokussieren,

10.1 Grundlegende Konzepte der Netzwerkanalyse

Netzwerke können auf unterschiedlichste Weise konzipiert werden, die dafür eingesetzten Begrifflichkeiten und Konzepte sind jedoch weitestgehend einheitlich. Nachfolgend werden zunächst aus graphentheoretischer Sicht die Bestandteile und Eigenschaften von Netzwerken beschrieben und einige Kennwerte zur Analyse von Netzwerken eingeführt. Beschrieben werden die Elemente und Maße anhand des Beispielnetzwerks in Abb. 10.1. Je nachdem, welche Bedeutung den Kreisen zugeschrieben wird, könnte das Netzwerk gemeinsam auftretende Wörter in Texten, Zugverbindungen zwischen Orten oder auch Freundschaftbeziehungen zwischen Menschen abbilden.

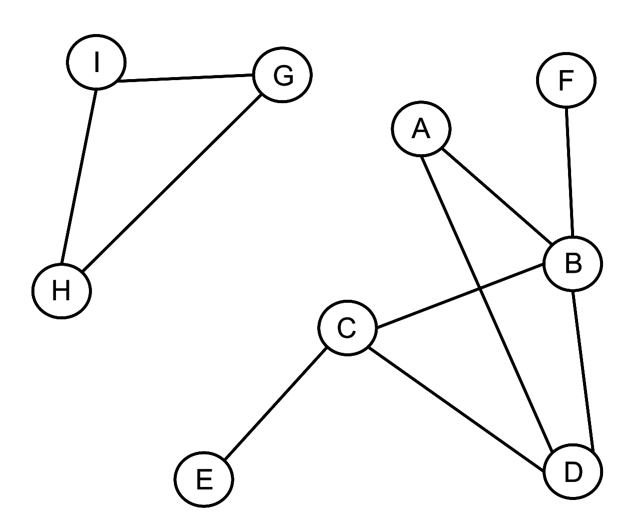


Abb. 10.1 Beispielnetzwerk. (Quelle: eigene Darstellung)

10.1.1 Elemente und Eigenschaften von Netzwerken

Netzwerke bestehen aus Akteuren oder Konzepten, die durch Beziehungen miteinander verbunden sind. Wenn die Kreise in Abb. 10.1 für Personen und die Linien zwischen ihnen für Freundschaften stehen, dann handelt es sich um ein Freundschaftsnetzwerk. Neben solch einer bildlichen Visualisierung von Netzwerken werden Netzwerke formal als Graph beschrieben. Ein Graph ist eine Menge von **Knoten** (die Menschen) und **Kanten** zwischen den Knoten (die Freundschaften). Es können zudem unterschiedliche Arten von Knoten in einem Netzwerk enthalten sein, etwa einerseits Menschen und andererseits die Geschäfte, in denen sie einkaufen. Netzwerke mit nur einer Art von Knoten heißen **unimodal**, wenn zwei Arten enthalten sind, spricht man von **bimodalen** oder bi-partiten Netzwerken, sind mehr als zwei Arten vorhanden, nennt man die Netzwerke **multimodal**.

Die Knoten und Kanten weisen unterschiedliche Eigenschaften auf. Wenn Menschen als Knoten aufgefasst werden, dann haben sie etwa soziodemografische Eigenschaften wie das Alter oder ein Geschlecht. Auf Ebene der Beziehungen kann man grundlegend die Stärke der Beziehungen, die Richtung sowie Reziprozität und die Multiplexität unterscheiden. Die **Stärke** gibt etwa an, wie häufig zwei Menschen miteinander in Kontakt kommen. Sie kann als sogenanntes Kantengewicht im Netzwerk angegeben werden. Spielt die **Richtung** keine Rolle, wie im Freundschaftsnetzwerk von Abb. 10.1, dann spricht man von ungerichteten Netzwerken, ansonsten von gerichteten. Das kann etwa auftreten, wenn man in jemanden verliebt ist, aber nicht zurückgeliebt wird. In Netzwerkabbildungen werden solche gerichteten Beziehungen durch Pfeile dargestellt, die in eine oder beide Richtungen weisen können. Liegen beide Richtungen vor, spricht man von reziproken oder symmetrischen Beziehungen. **Multiplexe** Beziehungen liegen vor, wenn mehrere Arten von Beziehungen gleichzeitig untersucht werden, beispielsweise die Freundschaft, der Umfang der Kommunikation und der Umfang gegenseitiger Unterstützung zwischen zwei Menschen.

In bimodalen Netzwerken, beispielsweise bestehend aus Personen und Veranstaltungen, können die Kanten darüber hinaus aus indirekten Beziehungen abgeleitet werden. Dazu unterstellt man, dass Personen, die auf der gleichen Party oder der gleichen Konferenz waren, mit einer gewissen Wahrscheinlichkeit in Beziehung zueinander stehen oder zumindest die Gemeinsamkeit aufweisen, am gleichen Ort gewesen zu sein. Umgekehrt geht man davon aus, dass eher keine Beziehung vorliegt, wenn sich zwei Menschen noch nie begegnet sind. Die gemeinsame Teilnahme wird dann als Beziehungsindikator gewertet, man spricht von **Affiliationsnetzwerken**. Auch Kooperationen können so erfasst werden, beispielsweise indem

Autor:innen, die zusammen Aufsätze oder Bücher publiziert haben, in Beziehung zueinander gesetzt werden. Das gleiche Verfahren lässt sich zur Konstruktion semantischer Netzwerke einsetzen. Zwischen Wörtern oder Konzepten, die häufig im gleichen Satz oder im gleichen Dokument auftreten, wird ein Zusammenhang unterstellt, sodass aus gemeinsamer Okkurrenz ein **Kookkurrenznetzwerk** entsteht (siehe Abschn. 9.1). Egal ob Kookkurrenz, Kooperation oder Affiliation, in allen Fällen werden aus den Verbindungen zwischen zwei Sorten von Knoten die Verbindungen zwischen einer Sorte von Knoten abgeleitet. Die Grundidee lässt sich vielfältig erweitern, indem man beliebige gemeinsame Eigenschaften als Verbindung begreift, etwa was Personen mögen und nicht mögen oder wo sie sich aufhalten und welche Orte sie meiden. Auf dieser Grundidee bauen Empfehlungssysteme von Onlineplattformen auf.

Durch die Verbindungen zwischen den Knoten entstehen innerhalb eines Netzwerks untereinander stark oder weniger stark verbundene Teilnetze, wobei sich mehrere Arten von Teilnetzwerken unterscheiden lassen:

- Der einfachste Fall besteht aus einer **Dyade**, das heißt, man betrachtet genau zwei Knoten und fragt danach, ob sie miteinander verbunden sind oder nicht.
- Das Konzept lässt sich auf drei Knoten erweitern, dann spricht man von **Triaden** – wie in der Abb. 10.1 zwischen G, H und I oder auch zwischen B, C und D. Auf dieser Grundlage lässt sich zum Beispiel untersuchen, inwiefern Freunde von Freunden auch Freunde sind.
- Wird die Bedingung, dass alle mit allen verbunden sein müssen, etwas gelockert, lassen sich über mehrere Ecken verbundene Teilnetze identifizieren. Je nach Verfahren spricht man von **Cliquen, Cores, Communities oder Komponenten** (zur Differenzierung siehe Jansen 2003, S. 193 ff.).
- Interessiert man sich nur für die Knoten und Beziehungen rund um einen einzelnen Knoten, dann spricht man von dem **Egonetzwerk** des Knotens. Ein Egonetzwerk erster Ordnung umfasst lediglich die direkt verbundenen Knoten, in zweiter bzw. höherer Ordnung werden auch die nachfolgenden Beziehungen zu weiteren Knoten erfasst.

10.1.2 Maße zur Analyse von (Teil-)Netzwerken

Um die Strukturen zwischen mehreren oder allen Knoten zu untersuchen und zu beschreiben, haben sich in der Netzwerkanalyse einige Maße etabliert (umfassend siehe Wasserman und Faust 1994):

- **Größe**: Zunächst kann man auszählen, wie viele Knoten ein Netzwerk umfasst. Das Gesamtnetzwerk aus Abb. 10.1 hat eine Größe von neun Knoten.
- **Dichte**: Über die Dichte wird angegeben, wie viele Beziehungen von allen möglichen Beziehungen tatsächlich realisiert sind.[2] Das Beispielnetzwerk weist eine Dichte von 0,28 auf, somit sind ein Drittel aller möglichen Beziehungen realisiert.
- **Reziprozität**: Weist ein Netzwerk gerichtete Beziehungen auf, kann über die Reziprozität angegeben werden, wie viele der Beziehungen ein- und wechselseitig sind.
- **Entfernung**: Wie viele Kanten zwischen zwei Knoten liegen, wird über die Pfadlänge angegeben. Um im Beispielnetzwerk von F zu E zu gelangen, benötigt man drei Schritte, von F zu B dagegen nur einen. Wie groß die Distanzen im gesamten Netzwerk sind, wird über die durchschnittliche Pfadlänge zwischen allen Knoten errechnet.[3]
- **Komponenten**: Die Anzahl der einzelnen Komponenten in einem Netzwerk zeigt, wie viele Teilnetzwerke untereinander in keiner Beziehung stehen. Im Beispielnetzwerk aus Abb. 10.1 finden sich zwei Komponenten (einmal die Knoten A bis F, dann die Knoten G bis I).

Ein wichtiges Konzept, um Netzwerke zu analysieren, ist die **Zentralität**. Zentralitätsmaße können zum einen für Netzwerke als Ganzes berechnet werden, um zu betrachten, wie stark alle Knoten von einigen wenigen Knoten abhängen. Netzwerke sind also nicht zwangsläufig flach, auch Hierarchien, Ketten oder Gitter lassen sich als Netzwerk darstellen. Hierarchische Beziehungen zwischen über- und untergeordneten Begriffen treten beispielsweise in semantischen Netzwerken auf. Zum anderen können Zentralitätsmaße auch für einzelne Knoten ermittelt werden, um Knoten in Schlüsselpositionen zu finden. Demnach ergeben sich die Eigenschaften von Akteuren und Konzepten aus der Netzwerkstruktur.

Um die Zentralität eines Knotens zu bestimmen, haben sich unterschiedliche Verfahren etabliert. Bevor Sie weiterlesen: Betrachten Sie einmal das Beispielnetzwerk (Abb. 10.1) und überlegen Sie, welchen Knoten Sie besonders wichtig finden und weshalb!

---

[2] Berechnet wird sie für gerichtete Netzwerke durch $\frac{\textit{Anzahl Kanten}}{\textit{Anzahl Knoten} \times (\textit{Anzahl Knoten} - 1)}$.

Für ungerichtete Netzwerke wird die Anzahl der Kanten doppelt gezählt.

[3] Dabei werden die jeweils kürzesten Pfade (sogenannte Geodäten, engl. *geodesics*) betrachtet.

Typische Zentralitätsmaße sind der Degree, die Betweenness und die Closeness:[4]

- Auf Knotenebene wird beim **Degree** die Anzahl der Beziehungen eines Knotens, etwa die Freunde im Freundschaftsnetzwerk, ausgezählt. Ein Knoten mit einem hohen Degree kann als ein populärer oder prestigeträchtiger Knoten interpretiert werden. So hat beispielsweise der Knoten B den höchsten Degree von 4, er hat also die meisten Beziehungen. Im Gegensatz dazu, kennen E und F je eine Person aus dem abgebildeten Netzwerk und haben damit einen Degree von 1.
- Ein Knoten kann auch dadurch eine zentrale Rolle spielen, dass er verschiedene Teilnetze verbindet, sodass viele Wege innerhalb des Netzwerks über ihn laufen. Er hat dann eine vermittelnde oder überbrückende Position. Möchte beispielsweise Knoten B den Knoten E kennenlernen, so könnte C die beiden miteinander bekannt machen. Ein solcher Knoten hat eine hohe **Betweenness**, ohne dass damit zwangsläufig ein hoher Degree einhergehen muss. Die Betweenness eines Knotens berechnet sich aus der Anzahl der kürzesten Pfade zwischen allen anderen Knoten, die über diesen Knoten laufen.
- Knoten sind auch dann zentral, wenn sie im Durchschnitt schnell alle anderen Knoten im Netzwerk erreichen. Diese indirekte Einbindung in das gesamte Netzwerk wird über die **Closeness** bestimmt. Sie berechnet sich entsprechend aus der durchschnittlichen Entfernung zu allen anderen Knoten. Im Freundschaftsnetzwerk können ebensolche Knoten schnell Informationen aus dem gesamten Netzwerk verbreiten oder erhalten.

Insgesamt können also unterschiedliche Analyseeinheiten in der Netzwerkanalyse herangezogen werden: die Knoten und Kanten jeweils für sich genommen, die verschiedenen Arten von Teilnetzwerken oder das Gesamtnetzwerk. Weil die Eigenschaften von Gesamtnetzwerken von den einzelnen Teilen abhängen und umgekehrt, spricht man von Emergenz: Das Ganze ist mehr als die Summe seiner Teile. Die Dichte des gesamten Netzwerks hängt von den Beziehungen zwischen einzelnen Akteuren ab, ohne dass die Akteure selbst schon eine Dichte hätten.

10.1.3 Hypothesentests und Netzwerkmodellierung

Das bislang vorgestellte Vokabular ist vor allem zur Beschreibung von Netzwerken geeignet. Netzwerkanalysen werden auch durchgeführt, um Zusammenhänge und Unterschiede zu erklären. So könnte man danach fragen, inwiefern das gleiche Ge-

[4] Für eine detaillierte Einführung in Zentralitätsmaße auf Knoten- und Netzwerkebene siehe beispielsweise Jansen (2003) und Freeman (1978).

schlecht oder gemeinsame Interessen verschiedener Personen dazu beitragen, dass sich Freundschaften ausbilden. Diese Fragestellungen lassen sich mit der klassischen Statistik nur eingeschränkt beantworten – zum einen, weil die Beobachtungen nicht unabhängig voneinander sind (eine grundlegende Annahme vieler statistischer Verfahren), und zum anderen, weil ein soziales Netzwerk immer schon typische Strukturmerkmale aufweist. So zeichnen sich Freundschaftsnetzwerke üblicherweise durch einen gewissen Anteil reziproker Beziehungen und lokaler Cluster aus. Auch ist in der Regel erwartbar, auf schiefe Degree-Verteilungen zu stoßen, das heißt, einige wenige Knoten sind deutlich stärker verbunden als die meisten anderen.[5]

Will man solche Aussagen (inferenz)statistisch überprüfen, so bieten sich Simulationen anstelle von klassischen Wahrscheinlichkeitsberechnungen an. Aus dem Vergleich von simulierten Welten mit der empirischen Welt lässt sich dann abschätzen, wo in der empirischen Welt überzufällige Zusammenhänge bestehen.[6] Um solche Zusammenhänge zwischen Eigenschaften von Knoten und Kanten unter Berücksichtigung struktureller Eigenschaften zu untersuchen, eignen sich beispielsweise Exponential Random Graph Models (Robins et al. 2007) oder Agentenbasierte Simulationen (siehe Kap. 11). Die Herausforderung besteht also darin, das Erwartbare vom Besonderen zu trennen.

10.1.4 Die Erhebung von Netzwerkdaten

Bei der Erhebung von Netzwerkdaten besteht das Ziel darin, Knoten und Kanten systematisch zu erfassen und in eine auswertbare Form zu bringen. Dabei sollte man sich vor Augen führen, dass Netzwerke nicht einfach vorliegen, sondern Beziehungen gezielt für die Datenanalyse konstruiert werden. Dafür können unterschiedliche Datenquellen herangezogen werden (Kap. 2). Prozessgenerierte Daten fallen unabhängig von wissenschaftlichen Projekten etwa bei der Nutzung von Onlineplattformen an und können teilweise über Webscraping oder Programmierschnittstellen (APIs) erhoben werden (siehe Kap. 7). Auch Datenbanken wie WikiData stellen eine Fundgrube für Netzwerkanalysen bereit, da die Daten bereits in einer relationalen Struktur erfasst werden (siehe Abschn. 3.7). Netzwerkanalytische Daten lassen sich zudem auch gezielt im Forschungsprozess über eigene Befragungen generieren. Sekundärdatenanalysen verwenden schließlich Daten, die in vorherigen Projekten erfasst wurden.

[5] Daraus ergibt sich ein interessanter Effekt: im Durchschnitt haben die eigenen Freund:innen mehr Freund:innen als man selbst (Feld 1991).

[6] Für eine Einführung in die Welt der Zufallsnetzwerke siehe Barabási (2016).

Einige Konstrukte, die Gegenstand sozial- oder geisteswissenschaftlicher Fragestellungen sind, verweisen unmittelbar auf Beziehungen, die direkt erhoben werden können. Solche expliziten Beziehungen werden zum Beispiel sichtbar, wenn sich Nutzer:innen auf sozialen Medien gegenseitig folgen oder liken. Darüber hinaus können Beziehungen auch indirekt abgeleitet werden. Wenn Nutzer:innen unter dem gleichen Post kommentieren, bauen sie nicht zwangsläufig bewusst eine Beziehung zueinander auf – allerdings kann ein Zusammenhang zwischen den Akteuren über das Kokommentieren konstruiert werden. Ebenso kann man bei gemeinsam auftretenden Wörtern in einem Text unterstellen, dass die räumliche Nähe auch eine semantische Nähe widerspiegelt, sodass sich daraus eine einfache Form semantischer Netzwerke konstruieren lässt (siehe Kap. 9). In der Netzwerkanalyse unterscheidet man deshalb zwischen einer realistischen und einer nominalistischen Perspektive. Erstere unterstellt, dass die untersuchten Netzwerke tatsächlich auch in der Wirklichkeit vorzufinden sind, während letztere von einem auf die jeweilige Fragestellung zugeschnittenen Konstruktionsprozess ausgeht (Laumann et al. 1983).

Je nach Umfang wird zwischen verschiedenen Erhebungsverfahren unterschieden (siehe auch Jansen 2003, Kap. 4). Bei einer **Vollerhebung** werden alle Knoten und Beziehungen eines Netzwerks erfasst. Man könnte beispielsweise eine Liste aller Mitglieder einer Universität erstellen und dann jede einzelne Person dazu befragen, welche anderen Personen sie kennt. Häufig stößt dieses Verfahren an praktische Grenzen. Eine Liste aller Webseiten gibt es beispielsweise nicht. Deshalb werden Sampling-Verfahren angewendet, um gezielt Netzwerkausschnitte zu erheben. Eine gängige Variante ist das Erheben von **egozentrierten Netzwerken**. Man beginnt bei einer Person oder Webseite und folgt dann schrittweise den Beziehungen. Je nachdem, wie viele Schritte man vom Ausgangspunkt weggeht, spricht man von Egonetzwerken der ersten, zweiten oder n-ten Ordnung. Genau dieses Verfahren wird von Suchmaschinen bzw. von den Webcrawlern der Suchmaschinen verwendet, um nach und nach alle Webseiten aufzufinden und in einer Datenbank abzuspeichern. Ego-Netzwerke sammeln in höheren Ordnungen oft schneeballmäßig eine große Zahl von Knoten, wodurch man bei der Verarbeitung dieser Daten schnell an Limitationen der verfügbaren Ressourcen stößt. Deswegen müssen häufig weitere Sampling-Entscheidungen getroffen werden, um möglichst systematisch und dennoch repräsentative oder zumindest informative Netzwerke zu erheben – beispielsweise wird für jeden Erhebungsschritt nur ein bestimmter Anteil von Knoten oder Kanten ausgewählt, der dann im nächsten Schritt weiterverfolgt wird (siehe zum Beispiel Leskovec und Faloutsos 2006 oder Salamanos et al. 2017).

Netzwerkdaten können durch Erhebung und Aufbereitung in unterschiedlichen Formen abgespeichert werden. Netzwerke lassen sich zunächst als **Matrizen** erfassen, bei denen Zeilen und Spalten die Knoten sind und eine Beziehung zwischen

Netzwerk als Matrix

	A	B	C	D	E	F	G	H	I
A	0	1	0	1	0	0	0	0	0
B	1	0	1	1	0	1	0	0	0
C	0	1	0	1	1	0	0	0	0
D	1	1	1	0	0	0	0	0	0
E	0	0	1	0	0	0	0	0	0
F	0	1	0	0	0	0	0	0	0
G	0	0	0	0	0	0	0	1	1
H	0	0	0	0	0	0	1	0	1
I	0	0	0	0	0	0	1	1	0

Netzwerke als Kantenliste

Quelle	Ziel
A	B
A	D
B	C
B	D
B	F
C	D
C	E
G	H
G	I
H	I

Abb. 10.2 Darstellung des Netzwerks aus Abb. 10.1 als Matrix und als Kantenliste. (Quelle: eigene Darstellung)

den Knoten durch 0 oder 1 in den Zellen markiert wird (siehe Abschn. 3.1; Abb. 10.2).[7] Die Diagonale der Matrix kann dazu verwendet werden, Beziehungen der Knoten zu sich selbst festzuhalten.[8] Bei ungerichteten Netzwerken, wie im Beispiel, sind die beiden Hälften oberhalb und unterhalb der Diagonalen identisch. Es macht dann also keinen Unterschied, ob man von der Zeile ausgehend die Spalte sucht oder umgekehrt. Bei gerichteten Netzwerken ist jeweils die eine Richtung (ausgehende Beziehungen) und die andere Richtung (eingehende Beziehungen) auf der unteren bzw. oberen Hälfte erfasst.

Soziale und semantische Netzwerke bestehen oft aus sehr vielen Knoten, wobei nur ein kleiner Anteil der möglichen Beziehungen realisiert ist. Matrizen enthalten deshalb häufig viele Nullen, sie sind nur spärlich besetzt (engl. *sparse*), was zu einer Platzverschwendung beim Abspeichern führt. Alternativ lassen sich Netzwerke

[7] Diese Form der Matrix wird auch als Adjazenzmatrix (engl. *adjacency matrix*) bezeichnet. Netzwerke können alternativ als Distanzmatrizen dargestellt werden, in denen die Zeilen und Spalten den Knoten entsprechen und in den Zellen die Pfadlängen – die Anzahl der Kanten, die zwischen zwei Knoten liegen – enthalten sind.

[8] Die Diagonale wird mitunter auch dazu verwendet, die Gesamtzahl der Beziehungen eines Knotens festzuhalten (Degree).

so erfassen, dass nur die bestehenden Beziehungen aufgelistet werden (Abb. 10.2). In der ersten Spalte einer solchen **Kantenliste** (engl. *adjacency list* oder *edge list*) wird die Quelle und in einer zweiten Spalte das Ziel aufgeführt. Die Stärke der Beziehung kann bei Bedarf in einer weiteren Spalte angegeben werden. Zusätzlich zur Liste aller Kanten wird gegebenenfalls eine **Knotenliste** erstellt, um weitere Eigenschaften zu erfassen, zum Beispiel neben einer Nummer für jeden Knoten auch eine Bezeichnung oder eine Kategorie.

Matrizen eignen sich gut, um bimodale Kookkurrenz- oder Affliationsnetzwerke (siehe oben) in unimodale Netzwerke umzuformen – etwa wenn das Auftreten von Wörtern in verschiedenen Texten in ein Netzwerk zwischen Wörtern umgewandelt oder aus dem gemeinsamen Besuch von Veranstaltungen eine soziale Beziehung abgeleitet werden soll. Stehen die Zeilen für Personen und die Spalten für Veranstaltungen bzw. die Zeilen für Dokumente und die Spalten für Wörter, kann man dies durch Matrixmultiplikation in ein unimodales Netzwerk umformen, in welchem es nur noch Personen oder Veranstaltungen bzw. Dokumente oder Wörter gibt.[9] Auch aus bimodalen Kantenlisten lassen sich unimodale Netzwerke erstellen. Wenn etwa Personen immer in der ersten Spalte aufgeführt sind und Veranstaltungen immer in der zweiten, zählt man aus, wie häufig bei immer zwei Personen die gleiche Veranstaltung angeführt ist. Diese Anzahl kann dann als Gewicht der Beziehung abgespeichert werden.[10]

10.1.5 Die Visualisierung von Netzwerken

Auch wenn soziale, semantische oder raumzeitliche Beziehungen mit unseren Sinnen nicht direkt wahrnehmbar sind, werden Netzwerke häufig durch visuelle Darstellungen erschlossen. Dabei gilt es zu beachten, dass die Visualisierung von Netzwerkdaten stets eine konstruierte Darstellung ist, das Bild eines Netzwerks ist

[9] Die Matrix wird mit der transponierten (= gedrehten) Matrix multipliziert. Je nachdem welche Matrix transponiert und ob links- oder rechtsmultipliziert wird, wird das Netzwerk zwischen den Zeilen oder zwischen den Spalten erzeugt (siehe Abschn. 4.2.4).

[10] Dabei entsteht zunächst ein ungerichtetes Netzwerk. Dieses lässt sich relativ unkompliziert in ein gerichtetes Netzwerk umrechnen, indem gemäß der Definition bedingter Wahrscheinlichkeiten die Anzahl gemeinsamen Auftretens an der Anzahl des Auftretens des einen Knotens standardisiert wird (siehe zum Beispiel van Atteveldt 2008). Die Beziehungen geben dann bedingte Wahrscheinlichkeiten an und lassen sich teilweise leichter interpretieren als absolute Häufigkeiten (siehe Kap. 9).

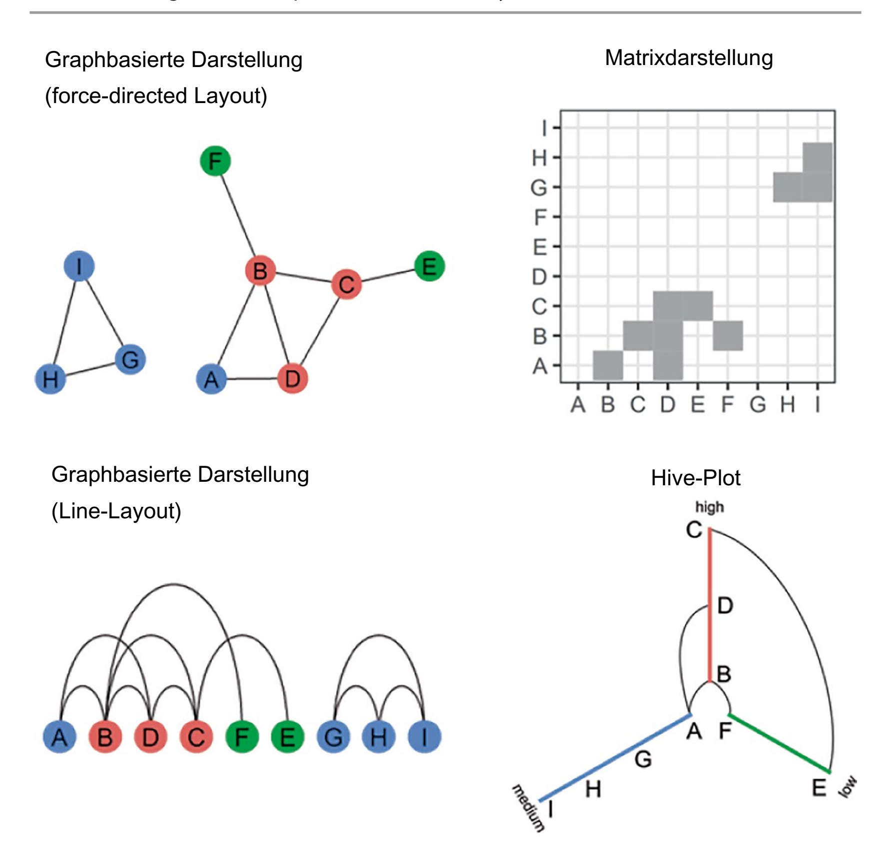


Abb. 10.3 Unterschiedliche Darstellungen des Beispielnetzwerks. Die Grafiken wurden mit *ggplot2* und *ggraph* in R erstellt (☛ *Repositorium*). Die Farben entsprechen dem Degree des jeweiligen Knotens: Knoten mit geringem Degree von null oder eins sind grün, diejenigen mit mittlerem Degree von zwei blau und Knoten mit hohem Degrees von drei oder mehr sind rot eingefärbt. (Quelle: eigene Darstellung)

nicht das Netzwerk selbst. Je nachdem, welche Aspekte eines Netzwerks betont werden sollen, eignen sich unterschiedliche bildliche Darstellungen (Abb. 10.3):

- Eine vordefinierte Anordnung ergibt sich aus **Matrixdarstellungen**. In einer visualisierten Adjazenzmatrix kann man erkennen, ob eine Beziehung vorhanden ist oder nicht. Wenn die Kanten ein Gewicht haben, können die Schnittpunkte aus Zeilen und Spalten auch eingefärbt werden, wodurch eine Heatmap entsteht.
- **Graphenorientierte Darstellungen** bilden alle Knoten ab, wobei die Kanten zwischen den Knoten als Linien visualisiert werden. Wichtig für die Interpretation ist die räumliche Anordnung (engl. *layout*) von Kanten und Knoten. Die

Knoten werden im einfachsten Fall in einer Reihe oder einem Kreis angeordnet und durch Linien oder Bögen verbunden. Hierarchische Netzwerke lassen sich auch gut als Bäume abbilden, um schnell über- und untergeordnete Knoten sichtbar zu machen. Sind die Netzwerke weniger klar geordnet, wird die Anordnung meist durch die Simulation physikalischer Kräfte zwischen den Knoten bestimmt. Eine Variante solcher force-directed Layouts stellen Spring-Embedder-Layouts dar (siehe zum Beispiel Fruchterman und Reingold 1991): Die Knoten stoßen sich durch simulierte elektrische Ladungen ab (*repulsion*), während sie gleichzeitig durch simulierte Zugfedern zusammengehalten werden (*attraction*). Lässt man eine entsprechende Simulation eine Zeitlang laufen, ordnen sich stark verbundene Knoten in unmittelbarer Nähe zueinander an. Solche Darstellungen sind in Programmen wie Gephi (Bastian et al. 2009) interaktiv implementiert, wodurch man in ein Netzwerk eintauchen und es explorieren kann. Nachteil solcher Abbildungen ist allerdings, dass Netzwerke schnell unübersichtlich werden, wenn sie groß sind. Sie sehen dann aus wie Hair Balls, aus denen man nur wenige nützliche Informationen herauslesen kann.
- Um strukturelle Eigenschaften großer Netzwerke übersichtlich zusammenzufassen, werden **Hive Plots** eingesetzt (Krzywinski et al. 2012). Auf den Achsen sind Werte wie der Degree abgebildet. Über die bestehenden Verbindungen zwischen den Achsen werden Eigenschaften des Netzwerks sichtbar. So kann beispielsweise schnell erkannt werden, welche Knoten degree-übergreifende Beziehungen aufbauen oder ob vielmehr eine Präferenz für andere Knoten mit einem ähnlichen Degree besteht (Assortativität bzw. Homophilie, siehe zum Beispiel Newman 2003).

10.2 Erhebung, Analyse und Visualisierung von Netzwerken

Im Folgenden wird ein Beispiel zur Erhebung, Analyse und Visualisierung von Netzwerkdaten durchgespielt. Das Verfolgen und Abspeichern von Beziehungen nennt sich Crawling. Ausgehend von einem YouTube-Video werden weitere, von YouTube empfohlene Videos erfasst. Es werden also direkt die auf YouTube durch das Empfehlungssystem implementierten Beziehungen verfolgt und als Netzwerk aufbereitet. Wie solche Empfehlungen zustande kommen, ist teilweise bei Covington et al. (2016) sowie Davidson et al. (2010) nachzulesen.

Das Beispiel demonstriert eine Kombination von Computational Methods, die sich auch auf andere Anwendungsfälle übertragen lässt. Für die Datenerhebung wird Facepager eingesetzt, um Daten über die API von YouTube zu erheben (siehe Abschn. 7.2). Zur Datenaufbereitung kommt die statistische Programmiersprache

R zum Einsatz (siehe Abschn. 5.1). Die Visualisierung findet schließlich mit Gephi statt, einem Tool für Netzwerkanalyse.

Für die Netzwerkanalyse findet sich eine Vielzahl nützlicher Werkzeuge, weitere Softwares sind etwa Neo4j (2022), RSiena (Snijders et al. 2021) oder Cytoscape (Shannon et al. 2003). Wichtig ist nicht so sehr, welches Tool verwendet wird – die Programme können, wenn Sie diesen Text lesen, schon längst anders aussehen oder sogar eingestellt worden sein –, sondern einen typischen Workflow nachzuvollziehen und dabei Anregungen für eigene Analysen zu gewinnen. Eine gute Anlaufstelle für weitere Software stellt die in der weiterführenden Literatur verlinkte Awesome List (Briatte 2021) dar.

10.2.1 Datenerhebung über die YouTube-API

Facepager ist ein Open-Source-Programm, mit dem ohne eigene Programmierung Daten über Programmierschnittstellen erhoben werden können. Das Programm ist vollständig in Python (siehe Abschn. 5.2) geschrieben und auf GitHub verfügbar.[11] Dort finden Sie auch ein Wiki mit kurzen Einführungen in verschiedene APIs (Getting Started). Die Parameter von APIs ändern sich immer wieder. Wenn Sie diesen Text lesen, müssen die folgenden Schritte möglicherweise bereits angepasst werden. Das Prinzip lässt sich aber auf andere APIs und Parameter übertragen, weitere Hinweise finden Sie dazu im ☛ *Repositorium* des Buchs. Installieren Sie zunächst eine aktuelle Version von Facepager.

Schritt 1: Startknoten hinzufügen Nach dem Starten von Facepager legen Sie mit dem Button New Database eine neue Datenbank an. Suchen Sie sich dann ein YouTube-Video als Startpunkt für das Netzwerk aus, zum Beispiel https://www.youtube.com/watch?v=4f9yC4ug8ZU. Der letzte Teil der URL, der auf `watch?v=` folgt, ist die eindeutige ID `4f9yC4ug8ZU` des Videos. Diese ID fügen Sie in Facepager über den Button Add Nodes in der Menü-Leiste als Startknoten (engl. *seed node*) ein.

Schritt 2: Ähnliche Videos abfragen Um empfohlene Videos abzufragen, kann das Preset „Get related videos“ verwendet werden, welches Sie über den Button Presets in der Kategorie „YouTube“ finden. In der Beschreibung des Presets erhalten Sie auch Hinweise zur Verwendung und insbesondere einen Link zur Dokumentation der API bei Google. Für den Moment können Sie das Preset einfach über Apply laden. Dabei werden die Voreinstellungen in das YouTube-Modul unten links übertragen (Abb. 10.4). Facepager setzt aus diesen Einstellungen eine URL zusammen, ruft diese URL auf und speichert das Ergebnis in einer Datenbank ab.

[11] Siehe Jünger und Keyling (2022; https://github.com/strohne/Facepager).

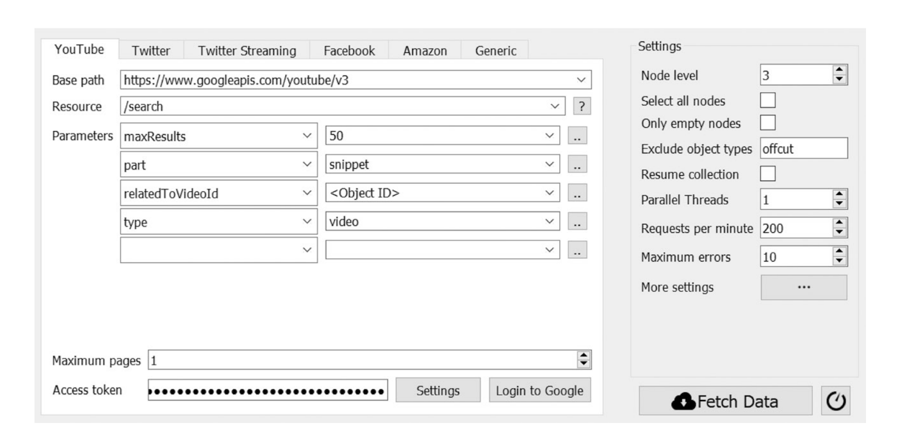


Abb. 10.4 Einstellungen in Facepager zur Erhebung von ähnlichen Videos. Hinweis: die Angabe des Node level auf der rechten Seite startet bei 1 und wird für jede Zone des Netzwerks um einen Schritt erhöht. (Quelle: eigene Darstellung)

Um die YouTube-API nutzen zu können, müssen Sie sich mit einem Google-Konto ausweisen. Zusätzlich muss das Konto mit einem eigenen Channel verbunden sein, den Sie ggf. direkt auf YouTube anlegen. Klicken Sie schließlich in Facepager auf den LOGIN-Button und loggen Sie sich ein. Das Passwort wird dabei nicht von Facepager abgefragt, sondern direkt von Google.[12] Facepager erhält anschließend ein sogenanntes Access Token, um sich gegenüber Google in Ihrem Namen auszuweisen. Wenn Sie das Feld mit dem Access Token später wieder leeren, ist keine weitere Anfrage möglich und Sie müssen sich bei Bedarf neu einloggen.

Nach dem Einloggen klicken Sie in der Nodes View – dem Bereich, in dem die einzelnen Datensätze dargestellt werden – den Startknoten „4f9yC4ug8ZU“ an und anschließend auf FETCH DATA. Das Ergebnis wird in der Datenansicht eingeblendet, ggf. müssen Sie den Knoten erst mit dem Dreieck links neben dem Knoten oder über EXPAND NODES aufklappen. In der Übersichtstabelle auf der linken Seite sind nur ausgewählte Daten angezeigt. Alle für einen Knoten abgefragten Daten, wie die Video-ID, das Veröffentlichungsdatum, den Titel oder die Videobeschreibung, sehen Sie in der Detailansicht auf der rechten Seite. Welche Spalten in der Tabelle erscheinen, wird über das Colum Setup rechts festgelegt.

Von diesem Egonetzwerk erster Ordnung können Sie nun weitergehen und die Videos der Videos abfragen. Sie müssen das nicht manuell für jeden einzelnen Knoten

[12] Das Verfahren nennt sich Open Authorization (OAuth 2.0) und ist unter anderem im Wiki von Facepager (Jünger 2020) erläutert.

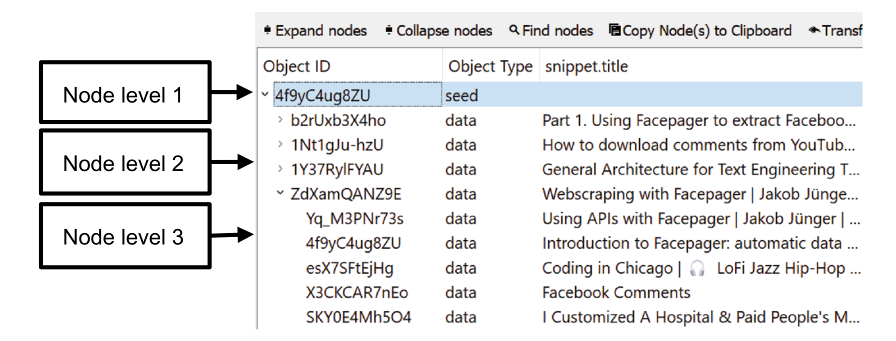


Abb. 10.5 Mit Facepager erhobene Videos. (Quelle: eigene Darstellung)

durchführen, Facepager unterstützt sie dabei. Wählen Sie wieder den Knoten „4f9y-C4ug8ZU" auf der obersten Ebene aus und erhöhen Sie in den Einstellungen das Node level (Abb. 10.4, rechts). Um die Videos der Videos abzufragen, stellen Sie das Node level auf 2 – da sich die abzufragenden Knoten auf der untergeordneten, zweiten Ebene des ersten Knotens befinden. Soll nach der zweiten Erhebung anschließend noch das Egonetzwerk der dritten Ordnung erhoben werden, setzen Sie das Node level anschließend auf 3 (Abb. 10.5). Sie können diese Schritte so lange wiederholen, wie Sie wollen, und die Ebene immer weiter erhöhen, müssen aber zunehmend mehr Zeit einplanen. Schon in der dritten Ebene sind im Beispiel über 2000 Knoten enthalten, sodass man schnell an die Rate Limits der API gerät (siehe Abschn. 7.2.3).[13]

Schritt 3: Daten exportieren und aufbereiten Wenn Sie die Datenerhebung abgeschlossen haben, können Sie die Daten über Export Data exportieren. Achten Sie darauf, dass in den Spalten alle Informationen enthalten sind, die für die Netzwerkanalyse und die Interpretation der Daten wichtig sind, wie die Namen der Kanäle und der empfohlenen Videos. Achten Sie in den Einstellungen des Exportfensters auch darauf, alle Knoten zu exportieren.

Die exportierte CSV-Datei können Sie zum Beispiel mit Excel öffnen, darin finden sich die in Facepager in der Übersicht angezeigten Daten. Zusätzlich ist je-

[13] Da zu dem Startknoten keine weiteren Informationen erhoben wurden, weist er im Export nur die ID auf. Um auch Angaben wie den „snippet.title" zu bekommen, fragen Sie vor der Erhebung der ähnlichen Videos mit dem YouTube-Preset „Get video statistics" die Details ab. Dadurch entsteht eine weitere Ebene, die Abfrage der ähnlichen Videos würde dann auf Node level 3 beginnen.

	id	parent_id	object_id	snippet.title
1	2	1	4f9yC4ug8ZU	Introduction to Facepager: automatic data collection using ...
2	4	2	d-nxFHYsxN0	Microsoft word tutorial |How to Insert Images into Word Do...
3	5	2	WrfHQKKowT0	Web scraping | Scrape eCommerce Websites Without Coding
4	6	2	gy4nUgPBHeM	Mining Twitter data for research: Part 1
5	7	2	b2rUxb3X4ho	Part 1. Using Facepager to extract Facebook Page posts and ...
6	8	2	lV9X2K8uEYE	How to create Data Entry Form in Excel - Ms Office?

Abb. 10.6 Hierarchischer Datensatz als Grundlage eines Netzwerks von ähnlichen Videos. Die Beziehungen können durch Abgleich von parent_id und id nachvollzogen werden. (Quelle: eigene Darstellung)

der Datensatz durch eine ID gekennzeichnet. Die Hierarchie zwischen den Datensätzen ist dadurch gekennzeichnet, dass im Feld „parent_id" die ID der übergeordneten Seite enthalten ist (Abb. 10.6). Das hat bereits Netzwerkcharakter – für die weitere Analyse erstellen Sie daraus eine Kanten- und eine Knotenliste. Die Daten müssen dazu so umgeformt werden, dass nicht die Beziehungen zwischen Datensätzen der Tabelle (von Facepager vergebene IDs), sondern zwischen den Videos (Video IDs bzw. Object IDs) abgebildet werden.

Die Kantenliste und eine Knotenliste lassen sich zum Beispiel mit R erzeugen. Im folgenden Beispiel wird davon ausgegangen, dass die Daten in der Datei *videos.export.csv* mit einem Semikolon als Trennzeichen im UTF8-BOM-Format abgespeichert wurden (☛ *Repositorium*):[14]

```
library(tidyverse)
videos <- read_csv2("videos.export.csv", na = "None")
```

Nach dem Einlesen in R werden zunächst mit `filter()` die relevanten Datensätze und über `select()` die nötigen Spalten ausgewählt. Neben den IDs zur Erfassung der Hierarchie werden die ID und der Name des Videos erfasst:[15]

[14] BOM steht für Byte Order Mark, siehe Abschn. 3.2. In Facepager lässt sich beim Exportieren wählen, ob eine BOM ausgegeben werden soll. Eine BOM ist normalerweise entbehrlich, erleichtert aber das Öffnen der Dateien mit Excel. Im Wiki von Facepager sind weitere Optionen für die Datenaufbereitung mit R aufgeführt.

[15] Die Datei enthält ggf. mehrere gleichbenannten ID-Spalten (einmal Facepager-IDs und einmal YouTube-IDs). In verschiedenen Package-Versionen werden diese durch `read_csv()` unterschiedlich behandelt, sodass Sie das Skript ggf. darauf anpassen müssen.

```
videos <- videos %>%
  filter(object_type == "data") %>%
  select(id, parent_id, object_id, snippet.title)
```

Daraus lässt sich nun eine Kantenliste gewinnen. Über einen `left_join` wird an jede Zeile die übergeordnete Zeile angehängt. Anschließend werden die relevanten Spalten ausgewählt und direkt in der `select()`-Funktion umbenannt. Abschließend werden Duplikate über `distinct()` und unvollständige Zeilen mittels `na.omit()` entfernt:

```
edges <- videos %>%
  left_join(videos, by = c("parent_id" = "id")) %>%
  select(source = object_id.y,
         target = object_id.x) %>%
  distinct()%>%
  na.omit()
```

Dadurch wurde jedem Video (als Ziel der Empfehlung) das übergeordnete Video (als Quelle der Empfehlung) zugeordnet. Die Kantenliste besteht dann nur noch aus zwei Spalten mit den IDs der Videos (Abb. 10.7).

In einer Knotenliste können zusätzlich zu den IDs die Namen der Videos oder weitere Merkmale wie der Kanalname festgehalten werden (Abb. 10.8). Da Videos mehrfach empfohlen werden können, entstehen bei der Erhebung Duplikate, die mit `distinct()` bereinigt werden sollten. Der Parameter `.keep_all` sorgt dafür, dass die anderen Spalten erhalten bleiben – die Angaben werden dann aus dem jeweils ersten Duplikat übernommen:

```
nodes <- videos %>%
  select(id = object_id, label = snippet.title) %>%
  distinct(id, .keep_all = T)
```

Um die so aufbereiteten Daten aufzubewahren oder in anderen Programmen weiterzuverarbeiten, können sie schließlich wieder als CSV-Dateien abgespeichert werden:

```
write_csv2(edges, "videos.edges.csv", na = "")
write_csv2(nodes, "videos.nodes.csv", na = "")
```

	source	target
1	4f9yC4ug8ZU	d-nxFHYsxN0
2	4f9yC4ug8ZU	WrfHQKKowT0
3	4f9yC4ug8ZU	gy4nUgPBHeM
4	4f9yC4ug8ZU	b2rUxb3X4ho
5	4f9yC4ug8ZU	lV9X2K8uEYE
6	4f9yC4ug8ZU	1Nt1gJu-hzU
7	4f9yC4ug8ZU	YAxLueEKqmU

Abb. 10.7 Kantenliste nach der Aufbereitung mit R. (Quelle: eigene Darstellung)

	id	label
1	4f9yC4ug8ZU	Introduction to Facepager: automatic data collection using ...
2	d-nxFHYsxN0	Microsoft word tutorial |How to Insert Images into Word Do...
3	WrfHQKKowT0	Web scraping | Scrape eCommerce Websites Without Coding
4	gy4nUgPBHeM	Mining Twitter data for research: Part 1
5	b2rUxb3X4ho	Part 1. Using Facepager to extract Facebook Page posts and ...
6	lV9X2K8uEYE	How to create Data Entry Form in Excel - Ms Office?
7	1Nt1gJu-hzU	How to download comments from YouTube with Facepager
8	YAxLueEKqmU	How Good Are Your Eyes? Cool and Quick Test

Abb. 10.8 Auszug aus der Knotenliste nach der Aufbereitung mit R. (Quelle: eigene Darstellung)

10.2.2 Statistische Analyse von Netzwerken

Auf diesem Datensatz können nun Netzwerkanalysen durchgeführt werden. Innerhalb von R stehen dafür Packages wie *igraph* zur Verfügung (Nepusz 2022). Die unterschiedlichen Netzwerk-Packages verwenden in der Regel eigene Datenstruk-

turen, um die Netzwerke zu verwalten. Als Brücke zwischen *igraph* und dem Tidyverse (siehe Kap. 5) bietet sich *tidygraph* an (Pedersen 2022). Mit nur einer Zeile lässt sich so aus der Knoten- und Kantenliste ein Netzwerkobjekt erzeugen:

```
library(igraph)
library(tidygraph)

graph <- tbl_graph(nodes, edges)
```

Sobald eine Knotenliste und eine Kantenliste vorliegen, eingelesen und in ein Netzwerkobjekt überführt wurden, kann das Netzwerk statistisch analysiert werden. Das *igraph*-Package hält dafür eine Vielzahl an Funktionen für alle Ebenen eines Netzwerks bereit (Tab. 10.1). Rufen Sie die Hilfe zum *igraph*-Package und zu den einzelnen Funktionen auf, um sich einen Überblick über die Möglichkeiten zu verschaffen.

Ein weiterer typischer Analyseschritt besteht darin, die zentralen Knoten zu bestimmen. Je nach Erkenntnisinteresse werden Zentralitätsmaße wie der Degree,

Tab. 10.1 Funktionen für die Netzwerkanalyse in R

Ebene	Befehl	Erläuterung
Gesamt-netzwerk	`print(graph)`	Größe des Netzwerks, das heißt die Anzahl der Knoten und Kanten
Gesamt-netzwerk	`count_components(graph)`	Anzahl der Komponenten
Gesamt-netzwerk	`graph.density(graph)`	Dichte des Netzwerks (= Anteil realisierter Beziehungen)
Gesamt-netzwerk	`transitivity(graph)`	Lokales Clustering im Netzwerk
Gesamt-netzwerk	`mean_distance(graph)`	Durchschnittliche Pfadlänge zwischen allen Knoten (= Erreichbarkeit)
Knoten	`degree_distribution(graph)`	Verteilung des Degrees der Knoten
Beziehungen	`dyad_census(graph)`	Anzahl einseitiger (engl. *asymetric*) und wechselseitiger (engl. *mutual*) Beziehungen
Teilnetze	`triad_census(graph)`	Anzahl der Triaden
Teilnetze	`max_cliques(graph, min=5)`	Größtmögliche Cliquen, die aus mindestens fünf Knoten bestehen

Einige der Kennwerte und Funktionen liegen für mehrere Analyseebenen vor. (Quelle: eigene Darstellung)

die Betweenness und die Closeness[16] verwendet. Mit den folgenden Funktionen aus dem *tidygraph*-Package werden diese Werte berechnet und im Netzwerkobjekt abgespeichert:

```
graph <- graph %>%
  activate("nodes") %>%
  mutate(degree = centrality_degree()) %>%
  mutate(betweenness = centrality_betweenness()) %>%
  mutate(closeness = centrality_closeness())
```

Die `activate()`-Funktion legt fest, ob die folgenden Operationen auf den Knoten oder den Kanten ausgeführt werden. Für die Interpretation kann die Knotenliste mit den neu berechneten Werten aus dem Netzwerkobjekt extrahiert werden. Dazu werden die Knoten aktiviert und in einen Dataframe (=ein Tibble) überführt:

```
nodes <- graph %>%
  activate("nodes") %>%
  as_tibble()
```

Anschließend lassen sich diese Daten mit typischen Funktionen aus dem Tidyverse exportieren, analysieren oder visualisieren. Geben Sie die Knotenliste beispielsweise mit der `arrange()`-Funktion nach den verschiedenen Zentralitätsmaßen (absteigend) sortiert aus:

```
nodes %>%
  arrange(-degree)

nodes %>%
  arrange(-betweenness)

nodes %>%
  arrange(-closeness)
```

[16] Bei der Berechnung der Closeness kann die Warnung auftreten: `closeness centrality is not well-defined for disconnected graphs`. Dies ist auf unverbundene Komponenten zurückzuführen. Da die Closeness die Entfernung eines Knoten zu allen anderen Knoten ermittelt und diese in unverbundenen Netzwerken unendlich ist, verwendet *igraph* als Alternative die größtmögliche Entfernung im Netzwerk, das heißt die *Anzahl aller Knoten* − 1. Bei der Interpretation der Werte ist das zu berücksichtigen.

Wenn Sie die Ergebnisse miteinander vergleichen, finden Sie eventuell Videos, die weniger prominent sind (geringer Degree), aber dennoch eine Schlüsselposition einnehmen und über die Nutzer:innen beim Verfolgen der Empfehlungen von einem Thema zu einem anderen gelangen (hohe Betweenness oder Closeness).

10.2.3 Visualisierung von Netzwerken

Sobald die Netzwerke in R eingelesen sind, können sie mit einem einfachen Befehl visualisiert werden. Bei großen Netzwerken kann man vorher mit `filter()` die Knoten mit einem kleineren Degree aussortieren, um die Grafik überschaubar zu halten:

```
graph <- filter(graph, degree > 1)
plot(graph)
```

Für schönere Grafiken lohnt sich ein Blick in das Package *ggraph* (Pedersen 2021). Interaktive Grafiken, zum Beispiel für die Einbettung in Webseiten, lassen sich dagegen mit dem Package *visNetwork* (Almende 2021) erzeugen (☛ *Repositorium*).

Für die Exploration von Netzwerken eignet sich insbesondere das Programm Gephi. Um die Daten dort weiterzuverarbeiten, benötigen Sie wie oben beschrieben jeweils eine CSV-Datei mit der Knotenliste und mit der Kantenliste. Sie können das Netzwerk auch zunächst mit R vorfiltern und eine übersichtlichere Knoten- und Kantenliste abspeichern. Bevor die Möglichkeiten zur Visualisierung von Netzwerken mit Gephi besprochen werden, ein Wort der Warnung: Stützen Sie Interpretationen nicht allein auf Grafiken. Netzwerkbilder helfen dabei, sich abstrakte Zusammenhänge besser vorzustellen. Es gibt jedoch so viele Möglichkeiten, dass kaum verbindliche und reproduzierbare Visualisierungen herstellbar sind. Versuchen Sie im Zweifelsfall, die Exploration der Bilder mit anderen Verfahren zu validieren. Die Visualisierung können Sie dazu nutzen, in der Analyse herausgearbeitete Erkenntnisse ansprechend darzustellen. Insbesondere die statistische Analyse liefert gut replizierbare Kennzahlen, mit denen die Eigenschaften eines Netzwerks auf den Punkt gebracht werden.

Schritt 1: Daten einlesen Gephi ist ein Programm, das für die Darstellung und Analyse umfangreicher Netzwerkdaten entwickelt wird. Laden Sie es von der Projektseite herunter und installieren Sie es auf Ihrem Computer.[17] Gephi ist in der Programmiersprache Java geschrieben, deshalb benötigen Sie, falls auf Ihrem

[17] Siehe Bastian et al. (2009; https://gephi.org/users/download/).

Computer noch nicht vorhanden, die Java-Laufzeitumgebung, achten Sie darauf die 64Bit-Version herunterzuladen.[18] Beim Start von Gephi werden Sie aufgefordert, ein Projekt zu öffnen oder ein neues anzulegen. Legen Sie zunächst ein neues Projekt an.

In Gephi lassen sich drei Bereiche unterscheiden, die über die Schaltflächen am oberen Fensterrand umgeschaltet werden:[19]

1. Im Overview werden die Optionen für die Visualisierung festgelegt (Farbe, Größe, Layout) und das Netzwerk dargestellt. Es können Daten gefiltert und Funktionen zur Berechnung von Kenndaten aufgerufen werden (Abb. 10.9).

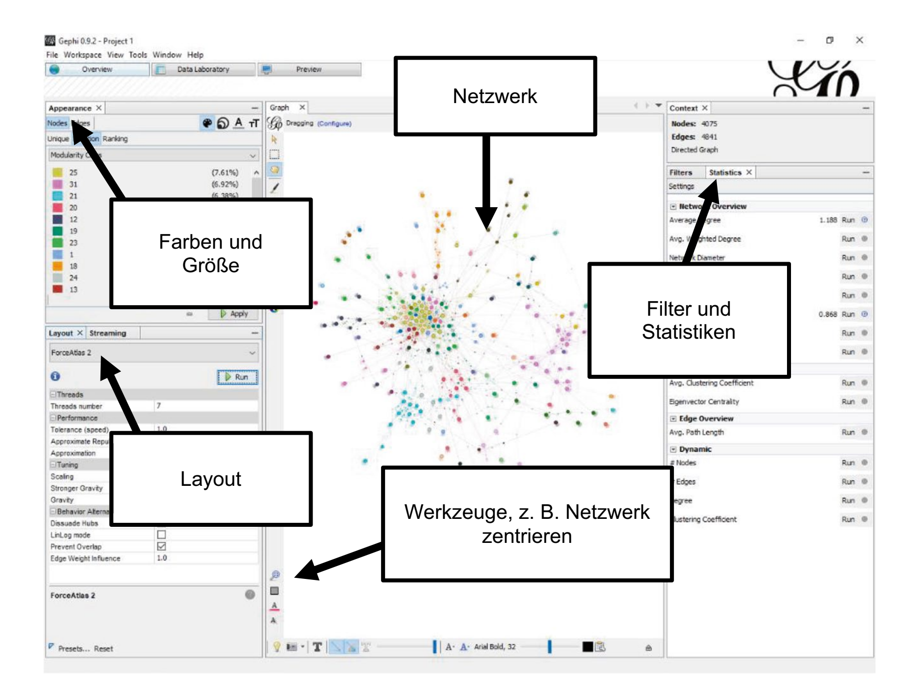


Abb. 10.9 Gephi im Überblick. (Quelle: eigene Darstellung)

[18] Zum Download siehe Oracle (2022a; https://www.java.com/de/download/).

[19] Für die Beispiele wird angenommen, dass die Benutzeroberfläche englischsprachig ist, stellen Sie die Sprache ggf. über den Menüpunkt Tools > Language auf Englisch um.

2. Im DATA LABORATORY werden die Kanten- und die Knotenliste aufgeführt. Hier lassen sich Daten importieren, filtern und bearbeiten.
3. Im PREVIEW-Fenster werden druckfähige Grafiken erstellt.

Wechseln Sie in den Bereich DATA LABORATORY und klicken Sie dort in der oberen Leiste auf IMPORT SPREADSHEET. Importieren Sie als erstes die Knotenliste und dann auf die gleiche Weise die Kantenliste. Achten Sie darauf, dass die Einstellung IMPORT AS jeweils auf „Nodes table" bzw. auf „Edges table" steht und hangeln Sie sich durch die Dialoge. Wichtig ist, dass Sie alle Daten in den gleichen Arbeitsbereich importieren. Sie müssen dazu unbedingt die Option APPEND TO EXISTING WORKSPACE auswählen (Abb. 10.10).

Für Gephi müssen in den importierten Dateien einige Konventionen eingehalten werden:

- In der Knotenliste muss es für jeden Knoten in der Spalte „id" eine eindeutige Kennung geben. Diese Kennung muss in der Kantenliste in den Spalten „source" und „target" zur Kennzeichnung der Beziehungen verwendet werden.

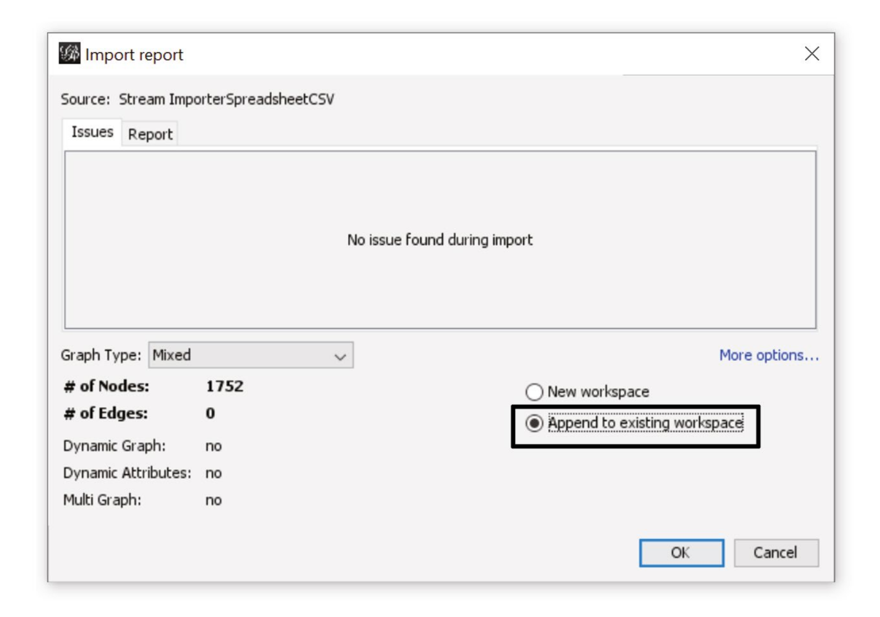


Abb. 10.10 Dialogfenster zum Import von Daten in Gephi. (Quelle: eigene Darstellung)

- Es sollten möglichst keine Knoten und keine Kanten doppelt vorkommen. Die Stärke von Beziehungen kann stattdessen numerisch über die Spalte „weight“ angegeben werden. Die Bezeichnung der Knoten wird in der Spalte „label“ abgelegt.
- Es können zusätzliche Spalten importiert werden, um zum Beispiel Knoten oder Kanten nach weiteren Merkmalen zu filtern oder grafisch unterschiedlich darzustellen.

Wenn Sie mit den oben erstellten Dateien *videos.nodes.csv* und *videos.edges.csv* arbeiten, dann können Sie die Voreinstellungen belassen.

Schritt 2: Die Knoten anordnen Zu Beginn sind die Knoten des Netzwerks zufällig verteilt. Je nach Zielstellung muss man sich zunächst für ein Layout entscheiden.[20] Folgende Schritte führen zu einer Darstellung, in der a) miteinander verbundene Knoten dichter beieinander sind als andere (force-directed layout), b) die Größe der Knoten durch die Anzahl der Beziehungen (degree) bestimmt wird und c) untereinander stark verbundene Bereiche durch eine gemeinsame Farbe gekennzeichnet werden (communities).

Wechseln Sie als Erstes in den Bereich OVERVIEW und wählen Sie im Abschnitt LAYOUT unter CHOOSE A LAYOUT den Algorithmus ForceAtlas 2 aus (Abb. 10.9). Dieser Algorithmus verwendet eine für umfangreiche Netzwerkdaten geeignete physikalische Simulation: Knoten stoßen sich grundsätzlich voneinander ab, die Kanten wirken aber wie Federn und ziehen die Knoten wieder zusammen (Jacomy et al. 2014). Klicken Sie auf RUN, um die Simulation zu starten und die Darstellung über die Parameter des Algorithmus optimieren:

- Verändern Sie das Scaling: Mit höheren Werten gehen die Knoten weiter auseinander.
- Verändern Sie die Gravity: Mit höheren Werten werden die Knoten stärker in die Mitte gezogen. Bei unverbundenen Graphen mit mehreren Komponenten hält eine starke Gravitation die Teilnetzwerke in der Mitte der Grafik.
- Wählen Sie abschließend PREVENT OVERLAP, damit die Knoten nicht übereinander liegen. Prüfen Sie, ob die Option LINLOG MODE die Darstellung verbessert.

[20] Die in Gephi vorhandenen Layout-Möglichkeiten lassen sich über Plugins erweitern.

Solange die Simulation läuft, können Sie einzelne Knoten mit der Maus verschieben und so die wirkenden Kräfte nachvollziehen. Wenn Sie das Netzwerk bei Ihren Versuchen aus den Augen verlieren, zentrieren Sie die Ansicht mit dem Lupen-Werkzeug (Abb. 10.9). Sobald Sie mit der Anordnung zufrieden sind, klicken Sie auf STOP.

Schritt 3: Größe, Farben und Beschriftungen Mit einer passenden Gestaltung der Knoten lassen sich die Eigenschaften des Netzwerks optisch schneller erfassen. In Gephi können vor allem Farbe und Größe der Knoten festgelegt werden. Als Grundlage können zum einen importierte Daten verwendet werden, zum Beispiel Kategorien. Es lassen sich zum anderen netzwerkanalytische Eigenschaften verwenden, die direkt mit Gephi berechnet werden. Öffnen Sie auf der rechten Seite den Abschnitt STATISTICS und berechnen Sie dort Average Degree und Modularity, indem Sie jeweils auf RUN klicken. Dabei wird für jeden Knoten der Degree berechnet und eine Zuordnung zu einem Cluster (Modularitätsklasse) vorgenommen. Das Ergebnis wird in die Datentabelle übernommen – schauen Sie im Bereich DATA LABORATORY nach!

Um die ermittelten Werte für die Visualisierung zu verwenden, wechseln Sie im Bereich OVERVIEW in den Abschnitt APPEARANCE (Abb. 10.11). Wählen Sie dort den Punkt NODES, klicken Sie auf die Schaltfläche für die Größe, wählen Sie RANKING entsprechend dem Degree**.** Ein Klick auf APPLY setzt die Änderungen um. Passen Sie die minimale und maximale Größe so an, dass Sie eine brauchbare Darstellung erreichen.

Im gleichen Bereich lässt sich auch die Farbe der Knoten auf die Modularitätsklasse einstellen (Abb. 10.12). Wählen Sie NODES und klicken Sie dieses Mal auf das Symbol für die Farben. Da die Clusterzugehörigkeit ein kategorisches und kein kontinuierliches Merkmal ist, wählen Sie PARTITION und stellen das Merkmal Modularity Class ein. Die Auswahl der Farben können Sie mit der Schaltfläche PALETTE verändern. Standardmäßig stehen nur acht unterschiedliche Farben zur Verfügung. Sie können die Anzahl erhöhen, indem Sie mit GENERATE eine neue Palette erzeugen (und ggf. die Option LIMIT NUMBER OF COLORS ausschalten). Mit einem Klick auf APPLY werden die Einstellungen übernommen.

Mit einem günstigen Layout und etwas Farbe lässt sich zwar die Gesamtstruktur eines Netzwerks überblicken. Um in die Details einzutauchen, muss man aber die Bedeutung der einzelnen Knoten kennen. Blenden Sie deshalb über die Symbolleiste unter dem Graphen die Label der Knoten ein (Abb. 10.13). Wenn Sie die

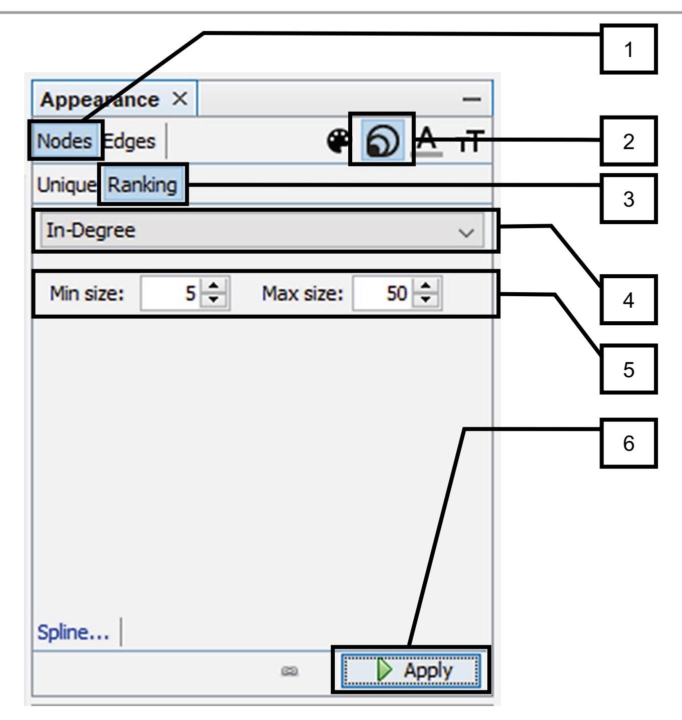


Abb. 10.11 Klickreihenfolge, um die Größe der Knoten am Degree auszurichten. (Quelle: eigene Darstellung)

Größe auf NODE SIZE einstellen, können Sie die Label mit dem Schieberegler an die Größe der Knoten anpassen.

Nun können Sie mit dem Scrollrad (oder auf dem Touchpad mit zwei Fingern) in das Netzwerk hineinzoomen und sich mit der Struktur vertraut machen. Mit der rechten Maustaste verschieben Sie den Ausschnitt. Sollten Sie verloren gehen, dann klicken Sie links unten in der Symbolleiste auf die Lupe, um die Darstellung in die Fenstergröße einzupassen.

Schritt 4: Mit Teilnetzwerken arbeiten Die Merkmale der Knoten und Kanten können nicht nur zur Visualisierung verwendet werden, sondern auch zum Redu-

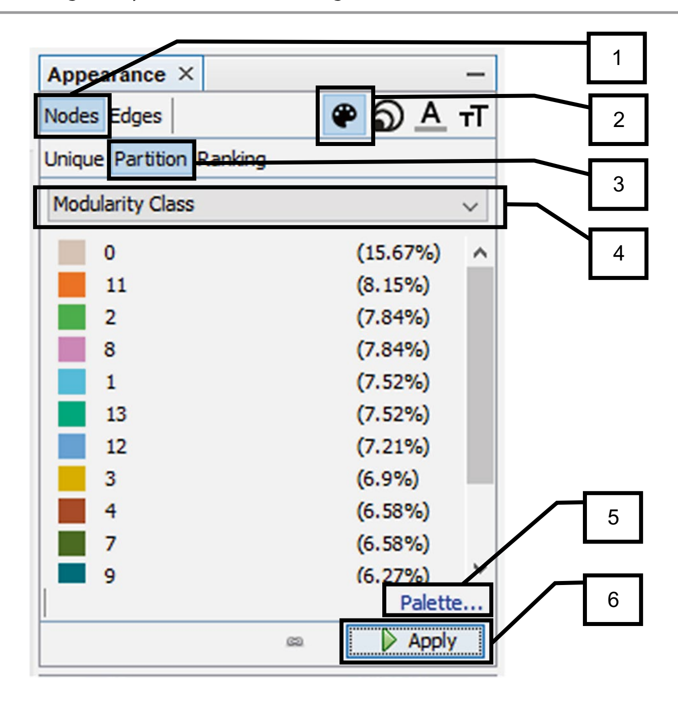


Abb. 10.12 Klickreihenfolge zum Einfärben der Knoten. (Quelle: eigene Darstellung)

zieren des Netzwerks, das heißt zum Herausarbeiten von Teilnetzen. Schauen Sie sich dazu den FILTER-Bereich auf der rechten Seite von Gephi genauer an:

- Kontinuierliche Eigenschaften wie der Degree von Nodes oder das Gewicht von Kanten werden über die unter ATTRIBUTES eingeordnete RANGE eingeschränkt. So können Sie das Netzwerk auf besonders stark verbundene Knoten eingrenzen.
- Kategorische Eigenschaften, zum Beispiel importierte Kategorien, lassen sich über ATTRIBUTES und anschließend PARTITION verwenden.
- Mit den Topologie-Filtern lässt sich die Ansicht auf Egonetzwerke oder untereinander stark verbundene Teilnetze einschränken – probieren Sie zum Beispiel k-Cores aus!

Die Filter werden mit der Maus per Drag & Drop in den QUERIES-Bereich gezogen. Mehrere Filter können als Subfilter hintereinandergeschaltet werden. Das Filtern wird mit der entsprechenden Schaltfläche aktiviert oder deaktiviert. Die so ausgewählten Teilnetze lassen sich schließlich in einen eigenen Arbeitsbereich ko-

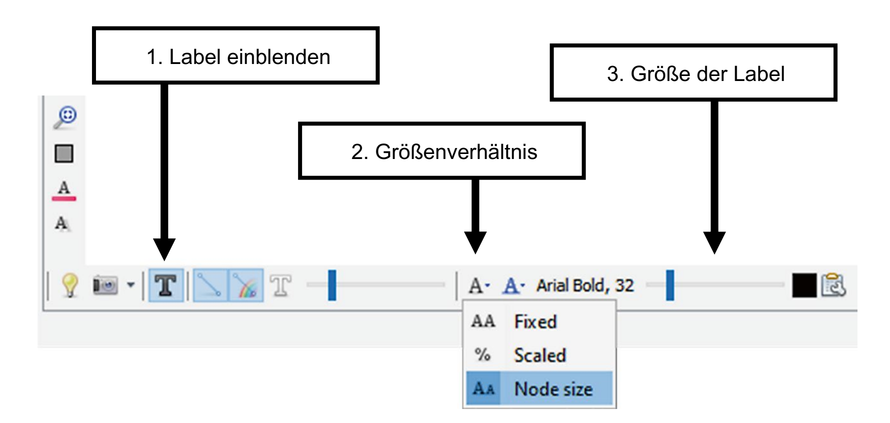


Abb. 10.13 Anpassen der Label. (Quelle: eigene Darstellung)

pieren und dort weiterverwenden. Dazu wählen Sie im DATA LABORATORY alle Knoten aus, klicken mit der rechten Maustaste auf einen der Knoten und wählen den Punkt COPY TO aus. Dort können Sie mit der weiteren Analyse und Visualisierung des Teilnetzwerks fortfahren.

10.2.4 Nächste Schritte

Netzwerkanalyse ist nicht einfach ein Methode unter vielen, sondern wendet sich der Welt mit einem spezifischen Blick auf die Beziehungen zwischen Akteuren, Konzepten und Ereignissen zu. Abstrahiert man von einzelnen Knoten und Kanten und betrachtet deren Einbettung in die Netzwerkstrukturen, lassen sich besondere Positionen und Eigenschaften herausarbeiten. So werden etwa strukturelle Unterschiede zwischen Vorgesetzten und Mitarbeiter:innen in Unternehmen sichtbar. Dabei kann man nicht nur einzelne Knoten und Kanten gegenüberstellen, sondern gesamte Netzwerke miteinander vergleichen – beispielsweise die Empfehlungswelten verschiedener Nutzer:innen. Auch Analysen im Zeitverlauf sind möglich, um die Evolution von Beziehungen und Strukturen zu erforschen.

Hier gibt es viel zu entdecken – die in diesem Kapitel angesprochenen Themen bewegen sich vorrangig auf der deskriptiven Ebene und schaffen dadurch eine Voraussetzung für weitere Explorationen. Die netzwerkanalytische Perspektive lässt sich auf nahezu beliebige Daten anwenden. Das gilt auch für Texte, wenn man die Kookkurrenz von Wörtern (siehe Kap. 9) als Netzwerke modelliert – versuchen Sie selbst einmal, auf diese Weise ein semantisches Netzwerk zu erstellen!

Übungsfragen

1. Was versteht man unter Knoten und Kanten?
2. Was ist der Unterschied zwischen den Maßen Degree, Betweenness und Closeness?
3. Was sagt eine Dichte von 0,6 über ein Netzwerk aus?
4. Stellen Sie sich vor, Sie wollen die Figuren aus Ihrem Lieblingsbuch als Netzwerk abbilden. Was können Sie als Kanten operationalisieren? Um welche Attribute könnten Sie Knoten und Kanten erweitern?
5. Sie haben ein Netzwerk in R konstruiert und wollen nun die zentralsten Knoten mit der Funktion `centrality_degree()` bestimmen. Weil Sie die Funktion noch nicht gut kennen, schlagen Sie in der Hilfe die möglichen Parameter nach. Wann geben Sie dabei den Parameter `directed = FALSE` an?
6. Wie werden Netzwerke in einem force-directed Layout angeordnet?
7. Warum sollten Sie Erkenntnisse nicht allein aus Netzwerkgrafiken ableiten und für welche Zwecke eignet sich die Visualisierung von Netzwerken?

Weiterführende Literatur

Barabási, A.-L. (2016). *Network Science*. Cambridge: Cambridge University Press.

Briatte. (2021). *Awesome Network Analysis. An awesome list of resources to construct, analyze and visualize network data*. Zugriff am 19.04.2022. https://github.com/briatte/awesome-network-analysis

Jansen, D. (2003). *Einführung in die Netzwerkanalyse. Grundlagen, Methoden, Forschungsbeispiele* (2., erw. Aufl.). Opladen: Leske+Budrich.

Luke, D. A. (2015). *A user's guide to network analysis in R.* Cham: Springer.

Wasserman, S. & Faust, K. (1994). *Social network analysis. Methods and application.* Cambridge: Cambridge University Press.

Literatur

Albrecht, S. (2013). Kommunikation als soziales Netzwerk? Anreize und Herausforderungen der Netzwerkanalyse von Kommunikationsprozessen. In B. Frank-Job, A. Mehler & T. Sutter (Hrsg.), *Die Dynamik sozialer und sprachlicher Netzwerke. Konzepte, Methoden und empirische Untersuchungen an Beispielen des WWW* (S. 23–46). Wiesbaden: Springer VS. https://doi.org/10.1007/978-3-531-93336-8_2

Almende B.V. and Contributors. (2021). visNetwork. Network Visualization using 'vis.js' Library (Version 2.1.0) [Computer software]. https://cran.r-project.org/package=visNetwork

Amaral, I. (2017). Computational Social Sciences. In L. A. Schintler & C. L. McNeely (Hrsg.), *Encyclopedia of Big Data* (S. 1–3). Cham: Springer. https://doi.org/10.1007/978-3-319-32010-6_41

Barabási, A.-L. (2016). *Network science*. Cambridge: Cambridge University Press. http://networksciencebook.com/

Bastian, M., Heymann, S. & Jacomy, M. 2009. *Gephi: An open source software for exploring and manipulating networks.* International AAAI Conference on Weblogs and Social Media. https://doi.org/10.13140/2.1.1341.1520

Beckert, J. (2005). Soziologische Netzwerkanalyse. In D. Kaesler (Hrsg.), *Aktuelle Theorien der Soziologie. Von Shmuel N. Eisenstadt bis zur Postmoderne* (S. 286–312). München: C.H. Beck.

Bourdieu, P. (1985). *Sozialer Raum und Klassen. Zwei Vorlesungen*. Frankfurt a. M.: Suhrkamp.

Briatte. (2021). *Awesome Network Analysis. An awesome list of resources to construct, analyze and visualize network data*. Zugriff am 19.04.2022. https://github.com/briatte/awesome-network-analysis

Cioffi-Revilla, C. (2010). Computational social science. *Wiley Interdisciplinary Reviews: Computational Statistics*, *2*(3), 259–271. https://doi.org/10.1002/wics.95

Covington, P., Adams, J. & Sargin, E. (2016). Deep Neural Networks for YouTube Recommendations. In S. Sen, W. Geyer, J. Freyne & P. Castells (Hrsg.), *Proceedings of the 10th ACM Conference on Recommender Systems* (S. 191–198). New York: Association for Computing Machinery (ACM). https://doi.org/10.1145/2959100.2959190

Davidson, J., Livingston, B., Sampath, D., Liebald, B., Liu, J., Nandy, P. et al. (2010). The YouTube video recommendation system. In X. Amatriain (Hrsg.), *Proceedings of the fourth ACM conference on Recommender systems* (S. 293–296). New York: Association for Computing Machinery (ACM). https://doi.org/10.1145/1864708.1864770

Diani, M. & McAdam, D. (2003). *Social Movements and Networks. Relational Approaches to Collective Action*. Oxford: Oxford University Press. https://doi.org/10.1093/0199251789.001.0001

Dijkstra, E. W. (1959). A note on two problems in connexion with graphs. *Numerische Mathematik*, *1*(1), 269–271. https://doi.org/10.1007/BF01386390

Euler, L. (2000). The seven bridges of Königsberg. In J. R. Newman (Hrsg.), *The world of mathematics* (Bd. 1, S. 573–580). Mineola: Dover Publications.

Feld, S. L. (1991). Why Your Friends Have More Friends Than You Do. *American Journal of Sociology*, *96*(6), 1464–1477. https://doi.org/10.1086/229693

Freeman, L. C. (1978). Centrality in social networks conceptual clarification. *Social Networks*, *1*(3), 215–239. https://doi.org/10.1016/0378-8733(78)90021-7

Fruchterman, T. M. J. & Reingold, E. M. (1991). Graph drawing by force-directed placement. *Software: Practice and Experience*, *21*(11), 1129–1164. https://doi.org/10.1002/spe.4380211102

Granovetter, M. S. (1973). The Strength of Weak Ties. *American Journal of Sociology*, *78*(6), 1360–1380. http://www.jstor.org/stable/2776392

Jacomy, M., Venturini, T., Heymann, S. & Bastian, M. (2014). ForceAtlas2, a continuous graph layout algorithm for handy network visualization designed for the Gephi software. *PloS One*, *9*(6), e98679. https://doi.org/10.1371/journal.pone.0098679

Jansen, D. (2003). *Einführung in die Netzwerkanalyse. Grundlagen, Methoden, Forschungsbeispiele* (2. Aufl.). Wiesbaden: Springer Fachmedien. https://doi.org/10.1007/978-3-663-09875-1

Jünger, J. (2020). *Facepager. Wiki*. Zugriff am 03.05.2022. https://github.com/strohne/Facepager/wiki

Jünger, J. & Keyling, T. (2022). Facepager. An application for automated data retrieval on the web. (Version 4.4.4) [Computer software]. https://github.com/strohne/Facepager/

Krzywinski, M., Birol, I., Jones, S. J. M. & Marra, M. A. (2012). Hive plots. Rational approach to visualizing networks. *Briefings in Bioinformatics*, *13*(5), 627–644. https://doi.org/10.1093/bib/bbr069

Latour, B. (1996). On actor-network theory. A few clarifications. *Soziale Welt*, *47*(4), 369–381. https://www.jstor.org/stable/40878163

Laumann, E. O., Marsden, P. V. & Prensky, D. (1983). The boundary specification problem in network analysis. In R. S. Burt & M. J. Minor (Hrsg.), *Applied network analysis. A methodological introduction* (S. 18–34). Beverly Hills: Sage.

Leskovec, J. & Faloutsos, C. (2006). Sampling from large graphs. In L. Ungar, M. Craven, D. Gunopulos & T. Eliassi-Rad (Hrsg.), *KDD '06: Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining* (S. 631–636). New York: Association for Computing Machinery (ACM). https://doi.org/10.1145/1150402.1150479

Malsch, T. & Schlieder, C. (2004). Communication without Agents? From Agent-Oriented to Communication-Oriented Modeling. In G. Lindemann, D. Moldt & M. Paolucci (Hrsg.), *Regulated Agent-Based Social Systems. First International Workshop, RASTA 2002, Bologna, Italy, July 16, 2002, Revised Selected and Invited Papers* (S. 113–133). Berlin: Springer. https://doi.org/10.1007/978-3-540-25867-4_7

Markov, A. A. (2006). An Example of Statistical Investigation of the Text Eugene Onegin Concerning the Connection of Samples in Chains. *Science in Context*, *19*(4), 591–600. https://doi.org/10.1017/S0269889706001074

Neo4j. (2022). Neo4j (Version 4.4.7) [Computer software]. https://neo4j.com/

Nepusz, T. (2022). igraph. Network Analysis and Visualization (Version 1.3.2) [Computer software]. https://cran.r-project.org/package=igraph

Newman, M. E. J. (2003). Mixing patterns in networks. *Physical Review*, *67*(2), 26126. https://doi.org/10.1103/PhysRevE.67.026126

Oracle. (2022a). *Get Java for desktop applications*. https://www.java.com/de/download/

Pedersen, T. L. (2021). ggraph. An Implementation of Grammar of Graphics for Graphs and Networks (Version 2.0.5) [Computer software]. https://cran.r-project.org/package=ggraph

Pedersen, T. L. (2022). tidygraph. A Tidy API for Graph Manipulation (Version 1.2.1) [Computer software]. https://cran.r-project.org/package=tidygraph

Quillian, M. R. (1967). Word concepts: a theory and simulation of some basic semantic capabilities. *Behavioral Science*, *12*(5), 410–430. https://doi.org/10.1002/bs.3830120511

Robins, G., Pattison, P., Kalish, Y. & Lusher, D. (2007). An introduction to exponential random graph (p*) models for social networks. *Social Networks*, *29*(2), 173–191. https://doi.org/10.1016/j.socnet.2006.08.002

Salamanos, N., Voudigari, E. & Yannakoudakis, E. J. (2017). Deterministic graph exploration for efficient graph sampling. *Social Network Analysis and Mining*, *7*, 24. https://doi.org/10.1007/s13278-017-0441-6

Schultz, F., Kleinnijenhuis, J., Oegema, D., Utz, S. & van Atteveldt, W. (2012). Strategic framing in the BP crisis. A semantic network analysis of associative frames. *Public Relations Review*, *38*(1), 97–107. https://doi.org/10.1016/j.pubrev.2011.08.003

Shannon, P., Markiel, A., Ozier, O., Baliga, N. S., Wang, J. T., Ramage, D. et al. (2003). Cytoscape: a software environment for integrated models of biomolecular interaction networks. *Genome Research*, *13*(11), 2498–2504. https://doi.org/10.1101/gr.1239303

Snijders, T. A. B., Ripley, R., Steglich, C., Koskinen, J., Niezink, N., Amati, V. et al. (2021). RSiena. Siena – Simulation Investigation for Empirical Network Analysis (Version 1.3) [Computer software]. https://cran.r-project.org/package=RSiena

Van Atteveldt, W. (2008). *Semantic network analysis. Techniques for extracting, representing and querying media content*. Charleston: BookSurge.

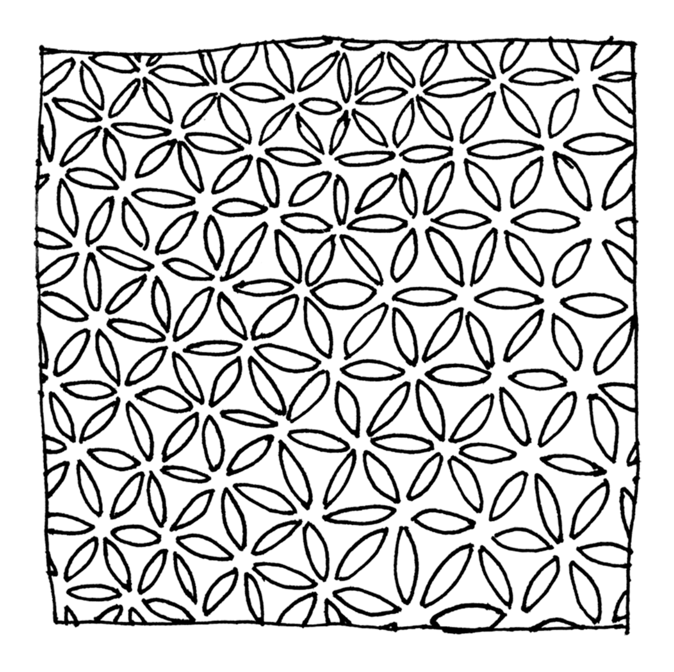
Wasserman, S. & Faust, K. (1994). *Social network analysis. Methods and applications*. Cambridge: Cambridge University Press.

White, H. C. (2008). *Identity and Control. How Social Formations Emerge* (2. Aufl.). Princeton: Princeton University Press.



Simulationsverfahren





J. Jünger, C. Gärtner, *Computational Methods für die Sozial- und Geisteswissenschaften*, https://doi.org/10.1007/978-3-658-37747-2_11

Zusammenfassung

Dieses Kapitel führt in Computersimulationen ein. Sie lernen die grundlegende Terminologie und verschiedene Arten von Computersimulationen kennen. Anhand eines Beispiels erkunden Sie, wie Agent-Based-Modeling umgesetzt wird.

Im Online-Repositorium unter https://github.com/strohne/cm finden Sie begleitend zum Kapitel weitere Materialien, auf die wir im Text mit ☛ verweisen.

Schlüsselwörter

Agentenbasierte Simulation · Monte-Carlo-Methode · Traceplot · Diffusionsmodelle · Mikro- und Makroebene · R

Das Ganze ist mehr als die Summe seiner Teile[1] – diese auf Aristoteles zurückgehende Feststellung gilt nicht nur für die im vorangegangenen Kapitel vorgestellte Netzwerkanalyse und verdeutlicht, dass sich viele soziale, geistige und kulturelle Phänomene aus dem Zusammenspiel komplexer Prozesse ergeben. Eine Geschichte besteht nicht einfach aus einer Aneinanderreihung von Ereignissen, vielmehr sind die Ereignisse so aufeinander bezogen, dass sich ein Spannungsbogen entwickelt. Auch soziale Systeme sind derart durch das wechselseitige Einwirken von Akteuren geprägt, dass sich daraus etwa die Polarisierung eines politischen Systems oder der öffentlichen Kommunikation ergeben kann. Will man solche Phänomene untersuchen, besteht die Herausforderung darin, Eigenschaften auf der Makroebene (Spannung, Polarisierung) durch Ereignisse auf der Mikroebene (Handlungen, Kommunikation) eines Systems zu erklären. In der wirklichen Welt sind die dahinterliegenden Prozesse in der Regel zu komplex, um sie vollständig zu erfassen – in durch Computersimulationen künstlich erzeugten Welten lassen sich dagegen gezielt einzelne Faktoren isolieren und untersuchen. Computersimulationen erlauben kontrafaktische Experimente. Damit ist gemeint, dass auch Ereignisse untersucht werden können, die in Wirklichkeit noch gar nicht eingetreten sind oder nicht eintreten werden. So ließe sich etwa für eine Videoplatt-

[1] „Das, was in der Weise zusammengesetzt ist, daß das Ganze Eines ist, ist nicht wie ein Haufen, sondern wie eine Silbe. Die Silbe ist aber nicht dasselbe wie ihre Buchstaben, BA ist nicht dasselbe wie B und A, ebenso Fleisch nicht dasselbe wie Feuer und Erde […]“ (Aristoteles 2013, S. 205).

form simulieren, welche Konsequenzen sich aus dem Ein- oder Abschalten des Empfehlungssystems für die Zugriffe auf einzelne Videos ergeben oder wie sich eine stärkere oder geringere Mitteilungsbereitschaft von Akteuren in der gesellschaftlichen Meinungsverteilung niederschlägt. Durch die notwendige Formalisierung von Prozessen zwingen Simulationsmodelle zu einer intensiven theoretischen Auseinandersetzung mit den untersuchten Phänomenen, erweitern dadurch das Verständnis und erlauben schließlich die Erklärung, Vorhersage und Erkundung komplexer Wirklichkeiten (Waldherr et al. 2021, S. 245 ff.).

Grundlegend werden zwei Arten von Simulationen unterschieden:

- **Agent-Based Modeling:** Bei agentenbasierten Simulationen (einführend zum Beispiel Gilbert und Troitzsch 2005) wird eine Welt mit Akteuren erzeugt, die sich im Zeitverlauf entwickelt. Die Akteure können etwa Menschen sein, die Nachrichten lesen und verbreiten. Dazu werden Regeln definiert, nach denen sich die Akteure verhalten. In jedem Schritt, das heißt in einer Epoche, führt jeder Akteur die festgelegten Aktionen aus. Diese Aktionen können etwa darin bestehen, sich zum einen in der Welt zu bewegen und zum anderen immer dann, wenn sich zwei Akteure treffen, Nachrichten weiterzugeben. Am Ende lässt sich beispielsweise auswerten, wie lange die Diffusion der Nachrichten bei unterschiedlichen Regelsätzen gebraucht hat. Zur Interpretation werden unter anderem Traceplots verwendet, auf denen Kennwerte wie die prozentuale Verbreitung einer Nachricht im Zeitverlauf dargestellt werden. Agentenbasierte Simulationen sind zwar abstrakt, wenn die Akteure aber in einer zweidimensionalen Welt platziert werden, lassen sie sich anschaulich wie in einem Computerspiel visualisieren.
- **Monte-Carlo-Simulationen:** Interessiert weniger der Zeitverlauf als das Ergebnis, so lassen sich Probleme als Monte-Carlo-Studien (einführend zum Beispiel Dunn und Shultis 2012) anlegen, deren Ergebnisse alternativ nur umständlich durch Formeln berechenbar wären. Ermittelt werden dabei die Wahrscheinlichkeiten, mit denen ein bestimmtes Ereignis eintritt. So lässt sich etwa bei verschiedenen politischen Wahlverfahren (z. B. Mehrheitswahl, Verhältniswahl) berechnen, mit welcher Wahrscheinlichkeit ein:e Kandidat:in in ein Gremium gewählt wird. Auch Verfahren wie das Topic Modeling (siehe Abschn. 8.2) basieren darauf, die wahrscheinlichste Zuordnung eines Themas zu einem Wort oder Text herauszufinden. Zunächst wird ein (hypothetischer) Datensatz mit den Ausgangsbedingungen definiert, aus dem dann viele Zufallsstichproben gezogen werden (engl. *resampling*). In Bezug auf politische Wahlen ergäbe sich die Wahlwahrscheinlichkeit, dass ein Ereignis ein-

tritt, dann aus der mittleren Wahrscheinlichkeit über viele Ziehungen hinweg. In Bezug auf die Themenstruktur beim Topic Modeling werden ebenfalls eine Vielzahl an möglichen Themenzuordnungen gezogen, um dann die insgesamt am häufigsten aufgetretene Zuordnung als den wahrscheinlichsten Fall anzunehmen.

Wenngleich beide Verfahren andere Perspektiven nahelegen, vereinen Simulationsstudien meist eine regelbasierte und eine wahrscheinlichkeitstheoretische Herangehensweise. Auch agentenbasierte Simulationen laufen nicht deterministisch ab: Welche Richtung ein Akteur auf dem Spielplan einschlägt oder welche Aktion ausgeführt wird, hängt wie bei Würfelspielen häufig von vorgegebenen Wahrscheinlichkeiten ab. Um den typischen Verlauf zu identifizieren, wird eine Vielzahl agentenbasierter Welten mit identischen Startbedingungen simuliert. Effektiv wird dadurch eine Stichprobe möglicher Welten gezogen und die Verteilung der resultierenden Kennwerte wird analysiert, um so etwa die Wahrscheinlichkeit für die Diffusion von Nachrichten zu schätzen. Besonders interessant werden solche Simulationen in Kombination mit Verfahren wie der Netzwerkanalyse, um die Diffusion von Wissen oder auch Viren in einer Gesellschaft zu untersuchen. Letztendlich ist es also eine Frage der Schwerpunktsetzung, für welches Verfahren man sich entscheidet.

In einem allgemeinen Sinn lassen sich alle zufallsbasierten Methoden als Simulationsverfahren begreifen. Auch die Signifikanztests der klassischen Statistik basieren darauf, die bei der Datenerhebung beobachteten Werte mit denjenigen Welten zu vergleichen, die sich bei zufällig aus einer Stichprobe gezogenen Werten ergeben. Dementsprechend finden sich eine Vielzahl an Algorithmen, Packages und Tools, mit denen sich Simulationen durchführen lassen. Zum Erlernen agentenbasierter Simulationen bietet sich beispielsweise Netlogo[2] an, das eine grafische Oberfläche, Beispielmodelle und eine einfach zu erlernende Programmiersprache bereitstellt. Simulationen können durchaus rechenintensiv sein, für die Kombination mit High Performance Computing (Abschn. 6.4) eignen sich Toolkits wie Repast.[3]

Das Grundprinzip einer Computersimulation lässt sich gut mit Skripten in Python (Abschn. 5.2) oder R (Abschn. 5.1) aufbauen und veranschaulichen. Stellen Sie sich dazu eine Welt vor, in der sich 100 Agenten zufällig bewegen. Einer dieser Agenten verbreitet Desinformationen und diese Informationen werden immer dann

[2] Siehe Wilensky (2021; https://ccl.northwestern.edu/netlogo/6.2.1/).

[3] Siehe Collier und North (2013; https://repast.github.io/).

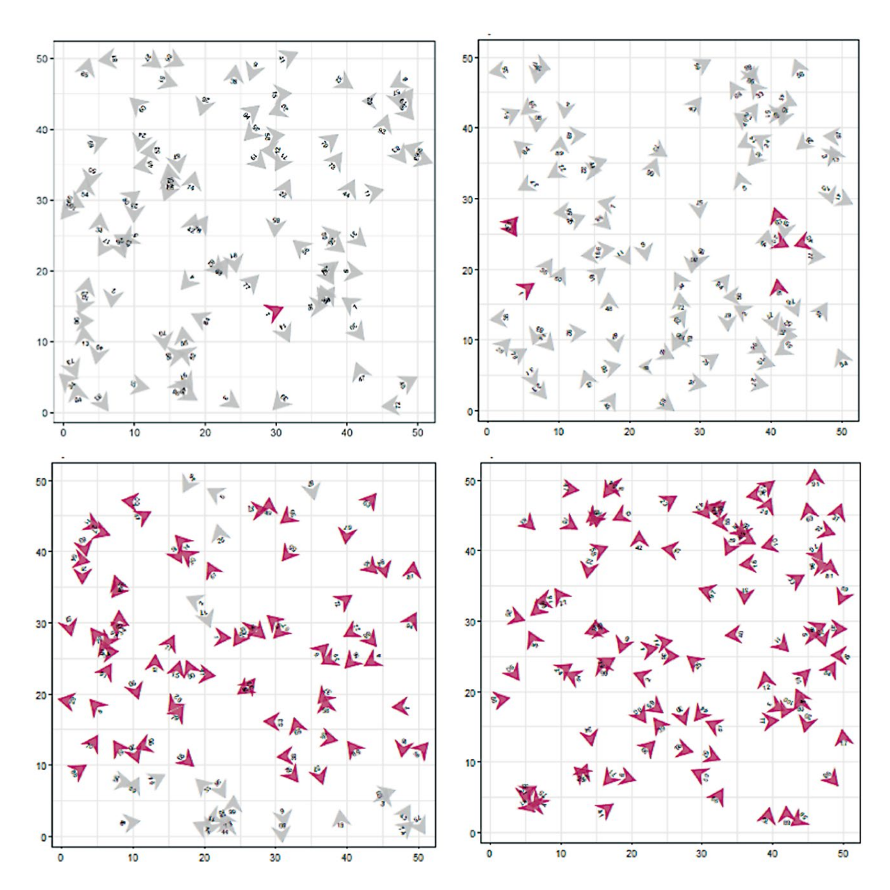


Abb. 11.1 Diffusion in einer künstlichen Welt. Die dunklen Agenten hatten bis zur jeweiligen Epoche mindestens eine Begegnung mit einem anderen dunklen Agenten (☛ *Repositorium*). (Quelle: eigene Darstellung)

weitergegeben, wenn sich zwei Agenten begegnen (Abb. 11.1). Mit diesem sehr einfachen Aufbau lässt sich zunächst modellieren, wie lange und in welcher Art und Weise ein Diffusionsprozess abläuft. Das Modell kann vielfältig interpretiert oder erweitert werden. So kann man dadurch etwa die Verbreitung von Wissen oder die Entstehung von Meinungsverteilungen erkunden. Komplexer werden solche Modelle, wenn verschiedene Themen in Konkurrenz zueinander stehen, Counterspeech eingebaut wird, die Akteure sich nicht zufällig bewegen, sondern untereinander vernetzt sind, oder einzelne Agenten in der Rolle von Journalist:innen eine Multiplikatorfunktion einnehmen.

11.1 Eine Welt entsteht

Eine Simulation lässt sich mit wenigen Schritten in R durchspielen. Legen Sie im ersten Schritt die Rahmenbedingungen der Welt fest, etwa die Breite und Höhe des Feldes, die Anzahl der Agenten auf diesem Feld und die Anzahl der Epochen, über die die Simulation laufen soll:

```
world_width <- 50
world_height <- 50
world_epochs <- 100
world_agents <- 100
```

Auf dieser Grundlage können nun einhundert Agenten zufällig in der Welt verteilt werden. Dazu erstellen Sie einen Dataframe (siehe Abschn. 5.1) mit den Spalten `no` für die von 1 an durchgezählte Nummer des Agenten, `x` sowie `y` für die Position auf dem Feld und `dir` für die Richtung zwischen 0 bis 359 Grad, in die der Agent blickt. In der Spalte `state` wird im Beispiel festgehalten, ob ein Agent die Nachricht wahrgenommen hat oder nicht.

```
agents <- data.frame(
  no  = c(1: world_agents),
  x   = sample(world_width, world_agents,replace=T),
  y   = sample(world_height, world_agents,replace=T),
  dir = sample(360, world_agents, replace=T) - 1,
  state = FALSE
)
```

Um die Agenten zufällig zu initialisieren, wird im Beispiel die Funktion `sample()` verwendet. Der erste Parameter gibt an, aus welchem Wertebereich eine Zahl zufällig gezogen werden soll, der zweite Parameter gibt die Anzahl der benötigten Werte an und der Parameter `replace=T` legt fest, dass auch mehrfach die gleiche Zahl gezogen werden darf, Agenten dürfen also die gleiche Position oder Ausrichtung haben.

Computer erzeugen in der Regel keine echten Zufallszahlen, sondern berechnen ausgehend von einem sogenannten Seed-Wert weitere Zahlen, die sich wie Zufallszahlen verhalten. Daraus ergibt sich der Vorteil, dass Zufallsziehungen und auch Simulationen reproduzierbar sind – setzen Sie vorab mit `set.seed(1852)` Ihre persönliche Lieblingszahl als Startwert!

Nachdem die Welt auf diese Weise erzeugt wurde, wird im Beispiel für den ersten Agenten der `state` auf `TRUE` gesetzt – dieser einzelne Agent beginnt mit der Verbreitung der Information.

```
agents$state[1] <-  TRUE
```

11.2 Die Welt entwickelt sich

Nun kann sich die Welt entwickeln. Um den Verlauf der Entwicklung festzuhalten, legen Sie einen leeren Dataframe `history` an. In einer Schleife, die die Epochen beginnend mit 1 hochzählt, wechseln sich anschließend zwei Vorgänge ab: Nachdem die Agenten ihre Aktionen ausgeführt haben, wird der aktuelle Zustand protokolliert. Im Beispiel wird im Dataframe `agents` die Nummer der Epoche abgelegt, um dann den kompletten Zustand mit `rbind()` an den bisherigen Verlauf anzuhängen. Dadurch wächst der Dataframe `history` mit jedem Durchgang stark an und kann so an die Grenzen des Arbeitsspeichers gelangen – bei umfangreicheren Simulationen würde man sich effizientere Verfahren ausdenken, etwa die Daten in einer Datei ablegen oder nur die wichtigsten Kennwerte der aktuellen Epoche protokollieren.

```
history <- data.frame()

for (epoch in c(1:world_epochs)) {

  print(epoch)

  # Aktionen ausführen
  agents <- agentsMove(agents)
  agents <- agentsChat(agents)

  # Protokollieren
  agents$epoch <- epoch
  history <- rbind(history, agents)
}
```

Die eigentlichen Aktionen sind hier in den Funktionen `agentsMove()` und `agentsChat()` gekapselt. Die erste Funktion nimmt den aktuellen Zustand der Welt entgegen, dreht jeden Agenten zufällig um maximal 25 Grad und bewegt ihn in die entsprechende Richtung. Wenn Sie die Mathematik dahinter nachvollziehen wollen, dann führen Sie die Zeilen schrittweise aus und verfolgen Sie die Ergebnisse:[4]

```
agentsMove <- function(agents) {

  # Zufällig zwischen -25 und +25 Grad rotieren
  rotate <- sample(c(-25:25),world_agents,replace=T)

  agents$dir <- agents$dir + rotate
  agents$dir <- agents$dir %% 360

  # Horizontal bewegen
  dir_x <- round(sin((agents$dir * pi) / 180))
  agents$x <- agents$x - dir_x
  agents$x <- (agents$x - 1) %% world_width + 1

  # Vertikal bewegen
  dir_y <- round(cos((agents$dir * pi) / 180))
  agents$y <- agents$y + dir_y
  agents$y <- (agents$y - 1) %% world_height + 1

  return (agents)
}
```

Die Funktion gibt den neuen Zustand zurück, damit im nächsten Schritt geprüft werden kann, ob ein Informationsaustausch stattfindet. Zunächst werden alle Informationsträger herausgefiltert. Anschließend erhalten alle Agenten den Zu-

[4] Um aus der Gradzahl eine Änderung in der Horizontalen oder Vertikalen zu berechnen, wird diese mit der Funktion $(dir \times \pi)/180$) in Radiant (rad) umgerechnet. Mittels Sinus und Cosinus wird daraus die Richtungsänderung berechnet. Die mehrfach eingesetzte Modulofunktion `%%` sorgt dafür, dass Agenten beim Erreichen des Spielfeldrandes oder bei einer Ausrichtung von höher als 359 Grad auf die 0 zurückspringen und so auf der anderen Seite wiederkehren, ganz als ob die Welt eine Kugel wäre.

stand `TRUE`, deren x- und y-Position mit mindestens einem Informanten übereinstimmen:[5]

```
agentsChat <- function(agents) {

  informants <- agents[agents$state == TRUE,]

  agents$state <-
    (agents$x %in% informants$x) &
    (agents$y %in% informants$y)

  return (agents)
}
```

Der Aufbau des Skripts wäre illustrativer möglich, dann aber weniger effizient, indem das Bewegen und Kommunizieren für jeden Agenten einzeln in einer Schleife (siehe Abschn. 5.1) durchgeführt würde – probieren Sie ruhig aus, die Funktionen entsprechend umzuschreiben oder das Beispiel in Python mit Loops nachzubauen!

11.3 Analyse der Ergebnisse

Da der Zustand der Welt in jedem Schritt protokolliert wurde, kann nicht nur das Resultat, sondern der gesamte Verlauf der Simulation nachvollzogen und visualisiert werden. Geht es um die Frage, wie schnell sich Informationen verbreiten, so lässt sich zu jedem Zeitpunkt der Anteil der Informationsträger an allen Agenten berechnen und im Zeitverlauf visualisieren (Abb. 11.2).

Zum Erstellen eines solchen Traceplots wird die Diffusionsrate für jede Epoche berechnet. Der Mittelwert der Variablen `state` entspricht dem Anteil der

[5] Der Ausdruck mit dem `%in%`-Operator gibt einen Boolean-Vektor zurück, der so lang ist wie die Anzahl der Agenten. Die Kombination der beiden Vektoren über `&` sorgt dafür, dass im Ergebnisvektor nur diejenigen Werte `TRUE` werden, auf die beide Bedingungen zutreffen. Das betrifft auch die Informanten selbst, die damit den Zustand `TRUE` behalten.

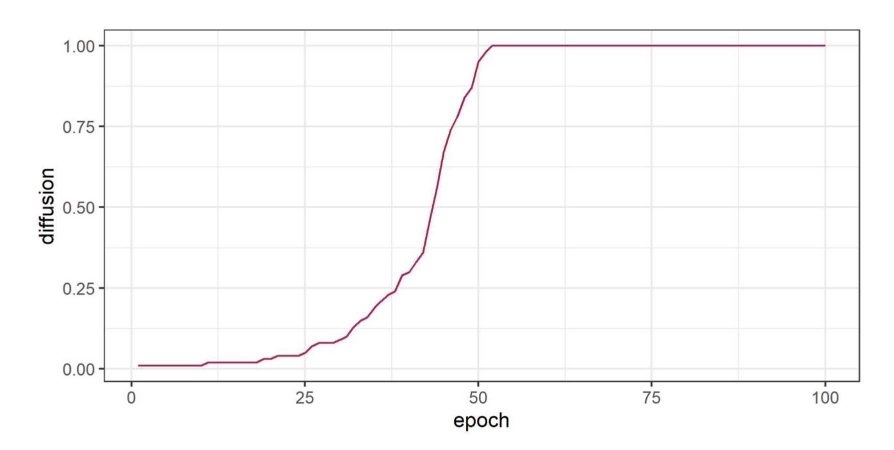


Abb. 11.2 Veränderung der Diffusion im Zeitverlauf (traceplot). (Quelle: eigene Darstellung)

`TRUE`-Werte, da Boolean-Werte bei der Berechnung automatisch zu 0/1 umkodiert werden:

```
trace <- history %>%
    group_by(epoch) %>%
    summarise(diffusion = mean(state)) %>%
    ungroup()
```

Das Ergebnis ist ein Dataframe mit den zwei Spalten `epoch` und `diffusion`. Mit *ggplot* kann daraus ein Liniendiagramm erstellt werden:

```
trace %>%
    ggplot(aes(epoch, diffusion)) +
    geom_line(color = "maroon")
```

Grundsätzlich bietet es sich an, alle Teilschritte der Simulation in eigene Funktionen auszulagern. So können leicht verschiedene Varianten der Diffusion ausprobiert werden. Da alle Epochen mitgeloggt wurden, können Sie dem Weltwissen beim Wachsen zuschauen und den Prozess sogar animieren (☛ *Repositorium*).

Man erkennt an den Skripten schnell, dass es sich um abstrakte Welten handelt. Das Handeln von Menschen wird über die Bearbeitung eines Datensatzes simuliert. Diese Art von Modellen findet sich häufiger in der Biologie, etwa um genetische Evolution zu analysieren. Evolutionsideen haben darüber hinaus auch Eingang in die Sozial- und Geisteswissenschaften gehalten, beispielsweise

wenn es um die Verbreitung von Memes (Dawkins 1976) oder Informationen geht. Das Ergebnis demonstriert einen Grundmechanismus der Informationsverbreitung und vieler anderer sozialer, geistiger, biologischer und physikalischer Prozesse: exponentielles Wachstum. Trotz der extrem geringen Ausgangswahrscheinlichkeit, dass zwei Agenten aufeinandertreffen und einer dieser beiden Agenten zudem noch die Information trägt, verbreitet sich die Information innerhalb von kurzer Zeit in der gesamten Welt. Schaffen Sie es, das exponentielle Wachstum mit veränderten Regeln (Vergessen), zusätzlichen Dingen (konkurrierende Mitteilungen, Netzwerkverbindungen) oder weiteren Eigenschaften der Agenten (Agilität, Wahrnehmungskapazität) und Mitteilungen (Neuigkeitswert) zu stoppen?

Übungsfragen

1. Wozu werden Simulationsmodelle eingesetzt?
2. Was unterscheidet agentenbasierte Simulationen von Monte-Carlo-Simulationen?
3. Was ist mit den Begriffen „Epoche“ und „Traceplot“ gemeint?
4. Installieren Sie Netlogo und implementieren Sie damit das im Kapitel vorgestellte Modell (Hinweis: in der Model Library finden Sie ein ähnliches Modell in der Kategorie *Biology/Evoplution/Genetic Drift/* unter dem Namen *GenDrift T interact*, das Sie entsprechend abwandeln können)!

Weiterführende Literatur

Dunn, W. L. & Shultis, J. K. (2012). *Exploring Monte Carlo methods.* Amsterdam: Academic Press.

Gilbert, G. N. & Troitzsch, K. G. (2005). *Simulation for the social scientist* (2. Aufl.). Maidenhead: Open University Press.

Snijders, T. A. (1996). Stochastic actor-oriented models for network change. *The Journal of Mathematical Sociology, 21*(1–2), 149–172. https://doi.org/10.1080/0022250X.1996.9990178.

Literatur

Aristoteles. (2013). *Metaphysik. Schriften zur ersten Philosophie*. Stuttgart: Reclam.

Collier, N. & North, M. (2013). Parallel agent-based simulation with Repast for High Performance Computing. *Simulation, 89*(10), 1215–1235. https://doi.org/10.1177/0037549712462620

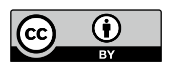
Dawkins, R. (1976). *The selfish gene*. New York: Oxford University Press.

Dunn, W. L. & Shultis, J. K. (2012). *Exploring Monte Carlo methods*. Amsterdam: Academic Press.

Gilbert, G. N. & Troitzsch, K. G. (2005). *Simulation for the social scientist* (2. Aufl.). Maidenhead: Open University Press.

Waldherr, A., Hilbert, M. & González-Bailón, S. (2021). Worlds of Agents: Prospects of Agent-Based Modeling for Communication Research. *Communication Methods and Measures*, *15*(4), 243–254. https://doi.org/10.1080/19312458.2021.1986478

Wilensky, U. (2021). NetLogo (Version 6.2.1) [Computer software]: Center for Connected Learning and Computer-Based Modeling, Northwestern University, Evanston, IL. https://ccl.northwestern.edu/netlogo/6.2.1/





12 Zusammenfassung

Zusammenfassung

Computational Methods bringen eine spezielle Perspektive auf die Welt mit sich. Sie sind einerseits ganz im Sinne der statistischen Tradition Werkzeuge, mit denen umfangreiche Daten erhoben, aufbereitet und analysiert werden, um so die Komplexität der Wirklichkeit auf wesentliche Aspekte zu reduzieren. Andererseits sind sie Praktiken, die sich zunehmend in den Alltag einweben, etwa durch die Etablierung sogenannter künstlicher Intelligenz in Anwendungsbereichen innerhalb und außerhalb der Wissenschaft. Das Kapitel gibt auf Basis einer Netzwerkanalyse Hinweise auf mögliche Lektürepfade, mit denen sich das vorliegende Buch erschließen lässt.

Schlüsselwörter

Computational Methods · Netzwerkanalyse · Lesehinweise

Computational Methods, wie sie in diesem Buch verstanden werden, bringen eine spezielle Perspektive auf die Welt mit sich. Sie sind einerseits ganz im Sinne der statistischen Tradition Werkzeuge, mit denen umfangreiche Daten erhoben, aufbereitet und analysiert werden, um so die Komplexität der Wirklichkeit auf wesentliche Aspekte zu reduzieren. Andererseits sind sie viel mehr als das: Praktiken, die sich zunehmend in den Alltag einweben, etwa durch die Etablierung sogenannter künstlicher Intelligenz in Anwendungsbereichen innerhalb und außerhalb der Wissenschaft. Dabei treten neue Komplexitäten auf und wer auf Arbeitserleichterung durch Automatisierung setzt, wird schnell enttäuscht. Nicht nur die Aufbereitung der Daten und das Debugging der Algorithmen sind zeitintensiv, sondern auch die

J. Jünger, C. Gärtner, *Computational Methods für die Sozial- und Geisteswissenschaften*, https://doi.org/10.1007/978-3-658-37747-2_12

Ergebnisse erfordern eine hermeneutische Auseinandersetzung mit Daten und Code (Jünger et al. 2022). Diese Beschäftigung ist allerdings lohnenswert und wenn Sie sich dafür entschieden haben, im vorliegenden Buch zu lesen, dann haben Sie im besten Fall viel Freude dabei, die Stringenz formaler Verfahren kreativ zu nutzen.

Was liegt aus dieser Perspektive näher, als die Zusammenfassung des Buchs automatisiert durch eine Maschine erstellen zu lassen? Die Basistechniken einer solchen Zusammenfassung sind in den einzelnen Kapiteln beschrieben. Im einfachsten Fall wählt man analog zum Extrahieren von Schlüsselwörtern (grundlegend siehe Turney 2000) besonders relevante Sätze aus und streicht alle anderen. Ein neu formulierter Text ließe sich daraus erstellen, indem die Sätze in eine semantische Repräsentation geparsed werden, aus der schließlich mit einem neuronalen Netz die Zusammenfassung generiert wird. Bis ein solches Encoder-Decoder-System überzeugende und spannende Ergebnisse produziert hat, ist allerdings viel Entwicklungsarbeit nötig, in dieser Zeit hat man das Buch mehrfach durchgelesen. Entsprechende Entwicklungsarbeit haben andere bereits geleistet und stellen öffentliche APIs zur Verfügung.[1]

Eine etwas andere, komprimierte Sicht auf das Buch ergibt sich, wenn man die Verweise zwischen den Kapitel nachverfolgt (Abb. 12.1). Auch für diese Art der Zusammenfassung sind die Basistechniken in den einzelnen Kapiteln zu finden. Irgendwie hängt alles mit allem zusammen, dennoch lassen sich hieraus instruktive Hinweise für die Lektüre gewinnen. Verfolgt man die Pfeile im Netzwerk rückwärts, ergeben sich mögliche Lesepfade. Zunächst wird deutlich, dass die beiden vorgestellten Programmiersprachen Python und R zu den zentralen Ausgangspunkten gehören, wobei wir je nach Anwendungsfall unterschiedliche Schwerpunkte gesetzt haben. Wollen Sie in die automatisierte Datenerhebung und speziell in das Webscraping einsteigen, dann dürfte es hilfreich sein, sich vorher die Grundlagen von Python anzueignen. In Bezug auf die verschiedenen Analyseverfahren, insbesondere die Textanalyse, setzen die Kapitel dagegen eher R voraus. Diese Einteilung hat sich aus der Arbeit in verschiedenen Forschungsprojekten heraus entwickelt, ist aber natürlich nicht zwingend und Sie werden vermutlich bei der Wahl der Sprachen eigene Vorlieben haben.

In jedem Fall empfiehlt es sich, die jeweils verknüpften Grundlagenkapitel zurate zu ziehen. Automatisierte Datenerhebung über Webscraping setzt ein Grundverständnis von HTML voraus und auch ein Überblick über Datenformate und

[1] Wenn Sie Techniken zur automatischen Texterzeugung ausprobieren wollen, lohnt sich ein Blick auf das Projekt GPT-3 (OpenAI 2022; https://openai.com/api/). Die Hintergründe des dort verwendeten Modells sind in Radford et al. (2022) beschrieben.

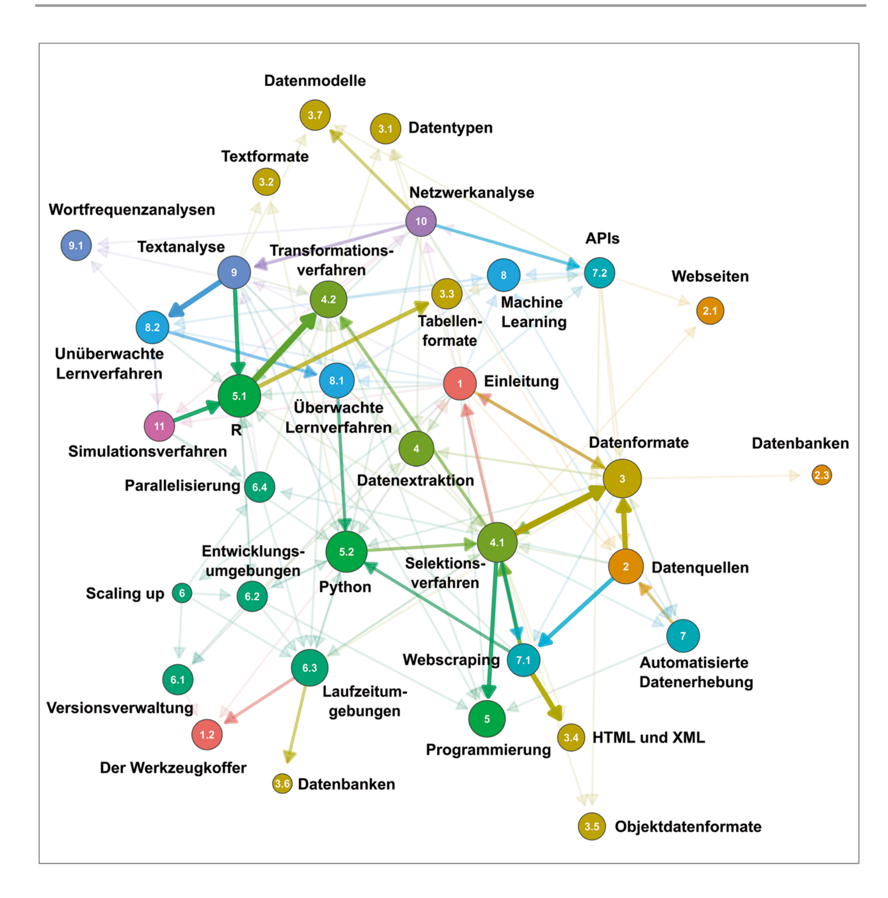


Abb. 12.1 Verweise zwischen den Kapiteln dieses Buchs. Die Kanten entsprechen Verweisen zwischen Kapiteln, die Kapitelnummer ist in den Knoten dargestellt. Die Hauptkapitel sind jeweils mit einer eigenen Farbe gekennzeichnet. Die Größe der Knoten und die Dicke der Kanten spiegeln die Anzahl der Verweise (Indegree) wider. (Quelle: eigene Darstellung)

Selektionsverfahren dürfte eine sinnvolle Ergänzung sein. Da bei der Datenanalyse häufig Daten umgeformt werden müssen, etwa vom Wide- in das Long-Format und wieder zurück, erscheint die Verknüpfung der R-Einführung mit dem Transformationskapitel naheliegend. Einige Themen bauen zudem durch die gewählten Beispiele aufeinander auf, so empfiehlt sich entsprechend der Kapitelreihenfolge erst ein Blick auf überwachte und anschließend auf unüberwachte Lernverfahren. Der einleitend vorgestellte Werkzeugkoffer wiederum enthält die für die Organisation

von Daten und Skripten unverzichtbare Kommandozeile. Sie werden Ihren eigenen Weg finden oder gefunden haben und dabei wünschen wir viel Vergnügen!

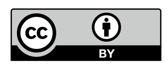
Literatur

Jünger, J., Geise, S. & Hänelt, M. (2022). Unboxing Computational Social Media Research From a Datahermeneutical Perspective: How Do Scholars Address the Tension Between Automation and Interpretation? *International Journal of Communication*, (16), 1482–1505.

OpenAI. (2022). *Build next-gen apps with OpenAI's powerful models*. Zugriff am 26.04.2022. https://openai.com/api/

Radford, A., Narasimhan, K. S., Salimans, T. & Sutskever, I. (2022). *Improving language understanding by generative pre-training*. Zugriff am 26.04.2022. https://cdn.openai.com/research-covers/language-unsupervised/language_understanding_paper.pdf

Turney, P. D. (2000). Learning Algorithms for Keyphrase Extraction. *Information Retrieval*, *2*(4), 303–336. https://doi.org/10.1023/A:1009976227802



Literatur

Abgeordnetenwatch.de. (2022). *Abgeordnetenwatch API Dokumentation*. Zugriff am 30.05.2022. https://www.abgeordnetenwatch.de/api
Ackoff, R. (1989). From data to wisdom. Presidential address to ISGSR, June 1988. *Journal of Applied Systems Analysis*, 16, 3–9.
Akademie der Wissenschaften und der Literatur Mainz. (2022). *Regesta Imperii*. Zugriff am 30.05.2022. http://www.regesta-imperii.de/
Albrecht, S. (2013). Kommunikation als soziales Netzwerk? Anreize und Herausforderungen der Netzwerkanalyse von Kommunikationsprozessen. In B. Frank-Job, A. Mehler & T. Sutter (Hrsg.), *Die Dynamik sozialer und sprachlicher Netzwerke. Konzepte, Methoden und empirische Untersuchungen an Beispielen des WWW* (S. 23–46). Wiesbaden: Springer VS. https://doi.org/10.1007/978-3-531-93336-8_2
Almende B.V. and Contributors. (2021). visNetwork. Network Visualization using 'vis.js' Library (Version 2.1.0) [Computer software]. https://cran.r-project.org/package=visNetwork
Almind, T. C. & Ingwersen, P. (1997). Informetric analyses on the world wide web: methodological approaches to 'webometrics'. *Journal of Documentation*, *53*(4), 404–426. https://doi.org/10.1108/EUM0000000007205
Alpine Linux Development Team. (2022). Alpine Linux (Version 3.15.4) [Computer software]. https://alpinelinux.org/
Amaral, I. (2017). Computational Social Sciences. In L. A. Schintler & C. L. McNeely (Hrsg.), *Encyclopedia of Big Data* (S. 1–3). Cham: Springer. https://doi.org/10.1007/978-3-319-32010-6_41
Amazon Mechanical Turk. (2018). *Amazon Mechanical Turk. Access a global, on-demand, 24x7 workforce*. Zugriff am 23.05.2022. https://www.mturk.com/
Amazon Web Services. (2022a). *Amazon Rekognition. Automatisieren Sie Ihre Bild- und Videoanalyse mit Machine Learning*. Zugriff am 30.05.2022. https://aws.amazon.com/de/rekognition/

J. Jünger, C. Gärtner, *Computational Methods für die Sozial- und Geisteswissenschaften*, https://doi.org/10.1007/978-3-658-37747-2

Amazon Web Services. (2022b). *AWS Lambda*. Zugriff am 03.05.2022. https://aws.amazon.com/de/lambda/

Anaconda. (2020). Anaconda Software Distribution [Computer software]. https://docs.anaconda.com/

Anaconda. (2022). Miniconda (Version 3) [Computer software]. https://docs.conda.io/en/latest/miniconda.html

AnsibleWorks. (2022). Ansible Automation Platform (Version 2.1) [Computer software]. https://www.ansible.com/

Anthony, L. (2022). AntConc. A freeware corpus analysis toolkit for concordancing and text analysis (Version 4.0.10) [Computer software]. https://www.laurenceanthony.net/software/antconc/

Apache Software Foundation. (2021). Apache Nutch (Version 1.18) [Computer software]. https://nutch.apache.org/

Apache Software Foundation. (2022a). Apache Hadoop (Version 3.3.3) [Computer software]. https://hadoop.apache.org/

Apache Software Foundation. (2022b). Apache HTTP Server (httpd) (Version 2.4.53) [Computer software]. https://httpd.apache.org/

Apify Technologies. (2022). Apify SDK. The scalable web crawling, scraping and automation library for JavaScript/Node.js. (Version 2.3.0) [Computer software]. https://sdk.apify.com/

Apogne. (2014). *Wait until page is loaded with Selenium WebDriver for Python,* Stack Overflow. Zugriff am 03.05.2022. https://stackoverflow.com/questions/26566799/wait-until-page-is-loaded-with-selenium-webdriver-for-python

Araujo, T., Ausloos, J., van Atteveldt, W., Loecherbach, F., Moeller, J., Ohme, J. et al. (2021). *OSD2F: An Open-Source Data Donation Framework*. https://doi.org/10.31235/osf.io/xjk6t

Aristoteles. (2013). *Metaphysik. Schriften zur ersten Philosophie*. Stuttgart: Reclam.

ATLAS.ti Scientific Software Development GmbH. (2022). ATLAS.ti (Version 22) [Computer software]. https://atlasti.com

Atlassian. (2022). *Become a git guru*. Zugriff am 25.04.2022. https://www.atlassian.com/git/tutorials

Attewell, P. A. & Monaghan, D. B. (2015). *Data mining for the social sciences. An introduction*. Oakland: University of California Press.

Ballsun-Stanton, B. (2010). Asking about data: Experimental philosophy of Information Technology. In *5th International Conference on Computer Sciences and Convergence Information Technology, ICCIT 2010* (S. 119–124). Piscataway: Institute of Electrical and Electronics Engineers (IEEE). https://doi.org/10.1109/ICCIT.2010.5711041

Barabási, A.-L. (2016). *Network science*. Cambridge: Cambridge University Press. http://networksciencebook.com/

Barbaresi, A. (2022). trafilatura: Web scraping tool for text discovery and retrieval (Version 1.2.1) [Computer software]. https://trafilatura.readthedocs.io/en/latest/

Bastian, M., Heymann, S. & Jacomy, M. 2009. *Gephi: An open source software for exploring and manipulating networks.* International AAAI Conference on Weblogs and Social Media. https://doi.org/10.13140/2.1.1341.1520

Bayer, M. (2012). Sqlalchemy. In G. Wilson & A. Brown (Hrsg.), *The Architecture of Open Source Applications. Volume II. Structure, scale and a few more fearless hacks.* Mountain View: aosabook.org.

BDZV. (2022). *Bundesverband Digitalpublisher und Zeitungsverleger*. Zugriff am 18.05.2022. https://www.bdzv.de/

Beck, K. (2006). *Computervermittelte Kommunikation im Internet*. München: Oldenbourg. https://doi.org/10.1524/9783486839203

Becker, A. (2022). HeidiSQL (Version 12.0) [Computer software]. www.heidisql.com/

Beckert, J. (2005). Soziologische Netzwerkanalyse. In D. Kaesler (Hrsg.), *Aktuelle Theorien der Soziologie. Von Shmuel N. Eisenstadt bis zur Postmoderne* (S. 286–312). München: C.H. Beck.

Beißwenger, M. (2013). Das Dortmunder Chat-Korpus. *Zeitschrift für germanistische Linguistik*, *41*(1), 161–164. https://doi.org/10.1515/zgl-2013-0009

Beißwenger, M., Fladrich, M., Imo, W. & Ziegler, E. (2020). Die Mobile Communication Database 2 (MoCoDa 2). In K. Marx, H. Lobin & A. Schmidt (Hrsg.), *Deutsch in Sozialen Medien* (S. 349–352). de Gruyter. https://doi.org/10.1515/9783110679885-018

Benoit, K. & Matsuo, A. (2020). spacyr. Wrapper to the 'spaCy' 'NLP' Library (Version 1.2.1) [Computer software]. https://cran.r-project.org/web/packages/spacyr/readme/README.html

Benoit, K., Muhr, D. & Watanabe, K. (2021). stopwords. Multilingual Stopword Lists (Version 2.3) [Computer software]. https://cran.r-project.org/package=stopwords

Benoit, K., Obeng, A., Watanabe, K., Matsuo, A., Nulty, P. & Müller, S. (2021). readtext. Import and Handling for Plain and Formatted Text Files (Version 0.81) [Computer software]. https://cran.r-project.org/package=readtext

Benoit, K., Watanabe, K., Wang, H., Nulty, P., Obeng, A., Müller, S. et al. (2018). quanteda: An R package for the quantitative analysis of textual data. *The Journal of Open Source Software*, *3*(30), 774. https://doi.org/10.21105/joss.00774

Benoit, K., Watanabe, K., Wang, H., Nulty, P., Obeng, A., Müller, S. et al. (2022). *quanteda: An R package for the quantitative analysis of textual data. Quick Start Guide*. Zugriff am 08.05.2022. https://quanteda.io/articles/quickstart.html

Berlin-Brandenburgische Akademie der Wissenschaften. (2022). *API (Schnittstellen zum DWDS)*. Zugriff am 30.05.2022. https://www.dwds.de/d/api

Berlind, D. (2022). *ProgrammableWeb*. Zugriff am 18.05.2022. https://www.programmableweb.com/

Björneborn, L. (2004). *Small-world link structures across an academic web space. A library and information science approach*. Copenhagen: Royal School of Library and Information Science.

Björneborn, L. & Ingwersen, P. (2004). Toward a basic framework for webometrics. *Journal of the American Society for Information Science and Technology*, *55*(14), 1216–1227. https://doi.org/10.1002/asi.20077

Blei, D. M. (2012). Probabilistic topic models. *Communications of the ACM*, *55*(4), 77–84. https://doi.org/10.1145/2133806.2133826

Blei, D. M. & Lafferty, J. D. (2007). A correlated topic model of Science. *The Annals of Applied Statistics*, *1*(1). https://doi.org/10.1214/07-AOAS114

Blei, D. M., Ng, A. Y. & Jordan, M. I. (2003). Latent Dirichlet Allocation. *Journal of Machine Learning Research*, *3*, 993–1022.

Boettinger, C., Eddelbuettel, D. & Ross, N. (2022). *The Rocker Project. Docker Containers for the R Environment*. Zugriff am 03.05.2022. https://www.rocker-project.org/

Bouchet-Valat, M. (2020). SnowballC: Snowball Stemmers Based on the C 'libstemmer' UTF-8 Library (Version 0.7.0) [Computer software]. https://cran.r-project.org/package=SnowballC

Bouma, G. (2009). Normalized (pointwise) mutual information in collocation extraction. In C. Chiarcos, R. E. de Castilho & M. Stede (Hrsg.), *Proceedings of the Biennial GSCL Conference 2009. From form to meaning: processing texts automatically* (S. 43–53). Tübingen: Narr.

Bourdieu, P. (1985). *Sozialer Raum und Klassen. Zwei Vorlesungen*. Frankfurt a. M.: Suhrkamp.

Bradski, G. (2000). The OpenCV library. *Dr. Dobb's Journal of Software Tools*. https://github.com/itseez/opencv

Brants, S., Dipper, S., Eisenberg, P., Hansen, S., König, E., Lezius, W. et al. (2004). TIGER: Linguistic Interpretation of a German Corpus. *Journal of Language and Computation*, *2004*(2), 597–620. https://www.ims.uni-stuttgart.de/forschung/ressourcen/korpora/tiger/

Bray, T. (Internet Engineering Task Force IETF.). (2014). *The JavaScript Object Notation (JSON) Data Interchange Format*. Zugriff am 24.05.2022. https://datatracker.ietf.org/doc/html/rfc7159

Briatte. (2021). *Awesome Network Analysis. An awesome list of resources to construct, analyze and visualize network data*. Zugriff am 19.04.2022. https://github.com/briatte/awesome-network-analysis

Brickley, D. & Miller, L. (2014). *FOAF Vocabulary Specification 0.99*. Zugriff am 24.05.2022. http://xmlns.com/foaf/spec/

Bruns, A. (2019). After the 'APIcalypse': social media platforms and their fight against critical scholarly research. *Information, Communication & Society*, *22*(11), 1544–1566. https://doi.org/10.1080/1369118X.2019.1637447

Burggraaff, C. & Trilling, D. (2020). Through a different gate: An automated content analysis of how online news and print news differ. *Journalism*, *21*(1), 112–129. https://doi.org/10.1177/1464884917716699

Burton, O. V. (2005). American Digital History. *Social Science Computer Review*, *23*(2), 206–220. https://doi.org/10.1177/0894439304273317

Busa, R. A. (2004). Perspectives on the Digital Humanities. In S. Schreibman, R. G. Siemens & J. Unsworth (Hrsg.), *A companion to digital humanities* (S. xvi–xxi). Oxford: Blackwell.

Bußmann, H. (1990). *Lexikon der Sprachwissenschaft*. Stuttgart: Kröner.

Canonical. (2022). Ubuntu [Computer software]. https://ubuntu.com/

Cao, L. (2017). Data Science. *ACM Computing Surveys*, *50*(3), 1–42. https://doi.org/10.1145/3076253

Center for Open Science. (2022). *Open Science Foundation*. Zugriff am 18.05.2022. https://osf.io/

Chambers, J. M. (2014). Object-Oriented Programming, Functional Programming and R. *Statistical Science*, *29*(2), 167–180. https://doi.org/10.1214/13-STS452

Chang, J. P., Chiam, C., Fu, L., Wang, A., Zhang, J. & Danescu-Niculescu-Mizil, C. (2020). *ConvoKit: A Toolkit for the Analysis of Conversations*. Proceedings of SIGDIAL. Zugriff am 30.05.2022. https://github.com/CornellNLP/ConvoKit

Chollet, F. (2020). Keras [Computer software]. https://keras.io/

Chomsky, N. (1956). Three models for the description of language. *IEEE Transactions on Information Theory*, *2*(3), 113–124. https://doi.org/10.1109/TIT.1956.1056813

CineStar. (2019). *Star Wars: Der Aufstieg Skywalkers*. Zugriff am 01.06.2022. https://www.cinestar.de/kino-greifswald/film/star-wars-episode-ix

Cioffi-Revilla, C. (2010). Computational social science. *Wiley Interdisciplinary Reviews: Computational Statistics*, *2*(3), 259–271. https://doi.org/10.1002/wics.95

Cioffi-Revilla, C. (2017). *Introduction to computational social science. Principles and applications* (2. Aufl.). London: Springer.

CKCEST: China Knowledge Centre for Engineering Sciences and Technology. (2022). *AMiner. Datasets for Social Network Analysis*. Zugriff am 30.05.2022. https://cn.aminer.org/data-sna

CLARIAH-DE. (2022). *Willkommen bei CLARIAH-DE*. Zugriff am 16.05.2022. https://www.clariah.de/

CLARIN. (2022). *CLARIN Virtual Language Observatory*. Zugriff am 21.06.2022. https://vlo.clarin.eu/

Clark, A. & Contributors. (2022). Pillow (Version 9.2.0) [Computer software]. *Zenodo*. https://doi.org/10.5281/zenodo.6788304

Cohen, D. J. & Rosenzweig, R. (2006). *Digital history. A guide to gathering, preserving, and presenting the past on the Web*. Philadelphia: University of Pennsylvania Press.

Coleman, J. S. (1964). *An Introduction to Mathematical Sociology*. New York: Free Press.

Collier, N. & North, M. (2013). Parallel agent-based simulation with Repast for High Performance Computing. *Simulation*, *89*(10), 1215–1235. https://doi.org/10.1177/0037549712462620

Cone, M. (2022). *Markdown Cheat Sheet. A quick reference to the Markdown syntax*. Zugriff am 24.05.2022. https://www.markdownguide.org/cheat-sheet/

Covington, P., Adams, J. & Sargin, E. (2016). Deep Neural Networks for YouTube Recommendations. In S. Sen, W. Geyer, J. Freyne & P. Castells (Hrsg.), *Proceedings of the 10th ACM Conference on Recommender Systems* (S. 191–198). New York: Association for Computing Machinery (ACM). https://doi.org/10.1145/2959100.2959190

Cox, M. & Ellsworth, D. (1997, 19. Oktober). Application-controlled demand paging for out-of-core visualization. In *Proceedings of the 8th conference on Visualization '97 (VIS '97)* (S. 235–244). Washington DC: IEEE Computer Society Press.

CRAN. (2022). *The Comprehensive R Archive Network*. Zugriff am 24.04.2022. https://cran.r-project.org/

Crossref. (2022). *Crossref Unified Resource API*. Zugriff am 03.05.2022. https://api.staging.crossref.org/swagger-ui/index.html

Dahley, D. (2017). *If statistics programs/languages were cars…* Zugriff am 11.07.2022. https://twitter.com/statsepi/status/872343239931682816

Datasette. (2022). *Openregister. Custom SQL query*. Zugriff am 30.05.2022. https://db.offeneregister.de/openregister

Davidson, J., Livingston, B., Sampath, D., Liebald, B., Liu, J., Nandy, P. et al. (2010). The YouTube video recommendation system. In X. Amatriain (Hrsg.), *Proceedings of the fourth ACM conference on Recommender systems* (S. 293–296). New York: Association for Computing Machinery (ACM). https://doi.org/10.1145/1864708.1864770

Dawkins, R. (1976). *The selfish gene*. New York: Oxford University Press.

DB Browser for SQLite. (2022). DB Browser for SQLite. The Official home of the DB Browser for SQLite [Computer software]. https://sqlitebrowser.org/

DBeaver. (2022). DBeaver. Free Universal Database Tool (Version 22.1.1) [Computer software]. https://dbeaver.io/

DBpedia Association. (2021). *DBpedia*. Zugriff am 24.05.2022. https://www.dbpedia.org/

DCMI: Dublin Core Metadata Innovation. (2022). *Dublin Core*. Zugriff am 30.05.2022. https://www.dublincore.org/

Derczynski, L., Vidgen, B., Kirk, H. R., Johansson, P., Chung, Y.-L., Guldborg, M., Kongsbak, K., Sprejer, L., & Zeinert, P. (2022). Hate Speech Dataset Catalogue. Zugriff am 05.01.2023. https://github.com/leondz/hatespeechdata

Destatis. (2022). *Statistisches Bundesamt*. Zugriff am 16.05.2022. https://www.destatis.de/

Deutsche Gesellschaft für Publizistik- und Kommunikationswissenschaft. (2022). *Meine Forschungssoftware*. Zugriff am 16.05.2022. https://www.dgpuk.de/de/forschungssoftware.html

Deutscher Wortschatz. (2022). *Wortschatz-Portal*. Zugriff am 08.05.2022. https://wortschatz.uni-leipzig.de/de

DHI. (2020). *The DHI and Digital Humanities*. Zugriff am 08.05.2020. https://www.ceu.edu/dhi/what-is-digital-humanities

Diani, M. & McAdam, D. (2003). *Social Movements and Networks. Relational Approaches to Collective Action*. Oxford: Oxford University Press. https://doi.org/10.1093/0199251789.001.0001

Digital Ocean. (2022). Scotch Box (Version 3.5) [Computer software]. https://github.com/scotch-io/scotch-box

Dijkstra, E. W. (1959). A note on two problems in connexion with graphs. *Numerische Mathematik*, *1*(1), 269–271. https://doi.org/10.1007/BF01386390

DiMaggio, P., Nag, M. & Blei, D. (2013). Exploiting affinities between topic modeling and the sociological perspective on culture: Application to newspaper coverage of U.S. government arts funding. *Poetics*, *41*(6), 570–606. https://doi.org/10.1016/j.poetic.2013.08.004

Docker (2021). *What can we help you find?* Zugriff am 25.04.2022. https://docs.docker.com/

Docker. (2022a). Docker Desktop (Version 4.7.1) [Computer software]. docker.com/products/docker-desktop/

Docker. (2022b). *Docker Hub*. Zugriff am 25.04.2022. https://hub.docker.com/

Docker. (2022c). *Reference documentation*. Zugriff am 05.07.2022. docs.docker.com/reference

Döring, N. & Bortz, J. (2016). *Forschungsmethoden und Evaluation in den Sozial- und Humanwissenschaften* (5. Aufl.). Berlin: Springer. https://doi.org/10.1007/978-3-642-41089-5

Dua, D. and Graff, C. (2019). *UCI Machine Learning Repository,* University of California, School of Information and Computer Science. Zugriff am 30.05.2022. https://archive.ics.uci.edu

Dunn, W. L. & Shultis, J. K. (2012). *Exploring Monte Carlo methods*. Amsterdam: Academic Press.

Dunning, T. (1993). Accurate Methods for the Statistics of Surprise and Coincidence. *Computational Linguistics*, *19*(1), 61–74.

Eclipse Foundation. (2022). Eclipse IDE. The Leading Open Platform for Professional Developers (Version 2022-06) [Computer software]. https://eclipseide.org/release/

Edmondson, M., Bryan, J., de Boer, J., Richardson, N., Kulp, D. & Cheng, J. (2022). googleAuthR: Authenticate and Create Google APIs (Version 2.0.0) [Computer software]. https://cran.r-project.org/package=googleAuthR

Elastic. (2022). Elasticsearch (Version 8.2) [Computer software]. https://www.elastic.co/elasticsearch

Entity Reconciliation Community Group. (2022). *Reconciliation Service API v0.1. A protocol for data matching on the Web*. Zugriff am 16.05.2022. https://reconciliation-api.github.io/specs/0.1/
Euler, L. (2000). The seven bridges of Königsberg. In J. R. Newman (Hrsg.), *The world of mathematics* (Bd. 1, S. 573–580). Mineola: Dover Publications.
Evolution Institute & Seshat Project. (2021). *Seshat: Global History Databank*. Zugriff am 30.05.2022. http://seshatdatabank.info/
Facebook. (2020). *The Open Graph protocol*. Zugriff am 02.07.2020. https://ogp.me/
Fandom. (2022). *Fandom. Search the world's largest fan wiki platform*. Zugriff am 16.05.2022. https://www.fandom.com/
Fayyad, U., Piatetsky-Shapiro, G. & Smyth, P. (1996). From Data Mining to Knowledge Discovery in Databases. *AI Magazine*, *17*(3), 37–54. https://doi.org/10.1609/aimag.v17i3.1230
Feasel, J. (2021). *SQL Fiddle*. Zugriff am 30.05.2022. http://sqlfiddle.com/
Feld, S. L. (1991). Why Your Friends Have More Friends Than You Do. *American Journal of Sociology*, *96*(6), 1464–1477. https://doi.org/10.1086/229693
Field, A., Miles, J. & Field, Z. (2012). *Discovering statistics using R*. Los Angeles: Sage.
Fielding, R. (2000). *Architectural styles and the design of network-based software architectures*. Dissertation. Irvine: University of California.
Fielding, R., Nottingham, M. & Reschke, J. (Internet Engineering Task Force (IETF), Hrsg.). (2022). *RFC 9110 HTTP Semantics*. https://www.rfc-editor.org/rfc/rfc9110.html
Fiesler, C., Beard, N. & Keegan, B. C. (2020). No Robots, Spiders, or Scrapers: Legal and Ethical Regulation of Data Collection Methods in Social Media Terms of Service. *Proceedings of the International AAAI Conference on Web and Social Media*, *14*(1), 187–196. https://ojs.aaai.org/index.php/ICWSM/article/view/7290
Firth, J. R. (1962). A synopsis of linguistic theory 1930–1955. In J. R. Firth, W. Haas, M. Halliday, W. Allen & R. H. Robins (Hrsg.), *Studies in linguistic analysis.* (S. 1–32). Oxford: Blackwell.
Freeman, L. C. (1978). Centrality in social networks conceptual clarification. *Social Networks*, *1*(3), 215–239. https://doi.org/10.1016/0378-8733(78)90021-7
Frey, J.-C. & Glaznieks, Aivars and Stemle, Egon W. (2019). *DIDI – The DiDi Corpus of South Tyrolean CMC 1.0.0,* Eurac Research CLARIN Centre. Zugriff am 30.05.2022. https://clarin.eurac.edu/repository/xmlui/handle/20.500.12124/7
Fruchterman, T. M. J. & Reingold, E. M. (1991). Graph drawing by force-directed placement. *Software: Practice and Experience*, *21*(11), 1129–1164. https://doi.org/10.1002/spe.4380211102
Fu, K. & Zhu, Y. (2020). Did the world overlook the media's early warning of COVID-19? *Journal of Risk Research*, 1–5. https://doi.org/10.1080/13669877.2020.1756380
Gentry, J. (2015). twitteR. R Based Twitter Client (Version 1.1.9) [Computer software]. https://cran.r-project.org/package=twitteR
GermEval. (2019). *GermEval Shared Task Hub. Natural Language Processing shared tasks for German*. Zugriff am 30.05.2022. https://germeval.github.io/
Gilbert, G. N. & Troitzsch, K. G. (2005). *Simulation for the social scientist* (2. Aufl.). Maidenhead: Open University Press.
Git. (2022a). Git. Fast-version-control (Version 2.36.0) [Computer software]. https://git-scm.com/downloads

Git. (2022b). *Documentation – Reference*. Zugriff am 25.04.2022. https://git-scm.com/docs
GitHub. (2022a). Atom (Version 1.60.0) [Computer software]. https://atom.io/
GitHub. (2022b). *Where the world builds software*. Zugriff am 25.04.2022. https://github.com/
GitHub. (2022c). GitHub Desktop (Version 2.9.15) [Computer software]. https://desktop.github.com/
GitLab. (2022). *The DevOps Platform has arrived.* Zugriff am 25.04.2022. https://about.gitlab.com/
Goldhahn, D., Eckart, T. & Quasthoff, U. (2012). Building Large Monolingual Dictionaries at the Leipzig Corpora Collection: From 100 to 200 Languages. *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC'12)*, 759–765. http://lrec-conf.org/proceedings/lrec2012/pdf/327_Paper
Golem. (2022). *IT-News für Profis*. Zugriff am 01.06.2022. https://www.golem.de/
Goodfellow, I. J., Dumitru, E., Carrier, P. L., Courville, A., Mirza, M., Hamner, B. et al. (2013). Challenges in Representation Learning: A report on three machine learning contests. *Neural Networks*, *64*, 59–63. https://doi.org/10.1016/j.neunet.2014.09.005
Google. (2014). *Public Data*. Zugriff am 30.05.2022. https://www.google.com/publicdata/directory
Google. (2022a). Cloud Bigtable [Computer software]. https://cloud.google.com/bigtable
Google. (2022b). *Google Search Central. Understand how structured data works*. Zugriff am 30.05.2022. https://developers.google.com/search/docs/advanced/structured-data/intro-structured-data
Google. (2022c). *Google Tabellen* [Computer software]. https://spreadsheets.google.com
Google. (2022d). *Cloud Vision API. Vision AI*. Zugriff am 30.05.2022. https://cloud.google.com/vision
Google. (2022e). *Cloud Vision-Preise*. Zugriff am 03.05.2022. https://cloud.google.com/vision/pricing
Google. (2022f). *Dataset Search*. Zugriff am 30.05.2022. https://datasetsearch.research.google.com/
Google. (2022g). *Google Cloud Platform*. Zugriff am 03.05.2022. https://console.cloud.google.com
Google. (2022h). TensorFlow. An end-to-end machine learning platform [Computer software]. https://www.tensorflow.org/
Google. (2022i). *Tutorials. Convolutional Neural Network (CNN)*. Zugriff am 08.05.2022. https://www.tensorflow.org/tutorials/images/cnn
Google, Microsoft, Yahoo & Yandex. (2022). *Schema.org*. Zugriff am 24.05.2022. https://schema.org/
Google Developers. (2022). *YouTube. Let users watch, find, and manage YouTube content*. Zugriff am 30.05.2022. https://developers.google.com/youtube/
GovData. (2022). *Das Datenportal für Deutschland. Open Government: Verwaltungsdaten transparent, offen und frei nutzbar*. Zugriff am 30.05.2022. https://www.govdata.de/
Goyvaerts, J. (2021). *Regular-Expressions.info. The Premier website about Regular Expressions*. Zugriff am 30.05.2022. https://regular-expressions.info/
Granovetter, M. S. (1973). The Strength of Weak Ties. *American Journal of Sociology*, *78*(6), 1360–1380. http://www.jstor.org/stable/2776392
Griffiths, T. L. & Steyvers, M. (2004). Finding scientific topics. *Proceedings of the National Academy of Sciences*, *101*(suppl_1), 5228–5235. https://doi.org/10.1073/pnas.0307752101

Grün, B. & Hornik, K. (2011). topicmodels : An R Package for Fitting Topic Models. *Journal of Statistical Software*, *40*(13). https://doi.org/10.18637/jss.v040.i13

Guy_Jantic. (2019). *If statistics programs/languages were cars…* Zugriff am 24.04.2022. https://www.reddit.com/r/rstats/comments/dn7fa7/i_thought_the_if_stats_programs_were_cars_meme/

Haim, M. (2020). Agent-based testing. An automated approach toward artificial reactions to human behavior. *Journalism Studies*, *21*(7), 895–911. https://doi.org/10.1080/1461670x.2019.1702892

Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D. [David] et al. (2020). Array programming with NumPy. *Nature*, *585*(7825), 357–362. https://doi.org/10.1038/s41586-020-2649-2.

Harrison, J. & Kim, J. Y. (2020). RSelenium. R Bindings for 'Selenium WebDriver' (Version 1.7.7) [Computer software]. https://cran.r-project.org/package=RSelenium

Harvard University. (2022a). *Harvard Dataverse*. Zugriff am 18.05.2022. https://dataverse.harvard.edu/

Harvard University. (2022b). *Social Science One,* Harvard's Institute for Quantitative Social Science. Zugriff am 23.05.2022. https://socialscience.one/

HashiCorp. (2022a). Vagrant (Version 2.2.19) [Computer software]. https://www.vagrantup.com/

HashiCorp. (2022b). *Vagrant Documentation*. Zugriff am 05.07.2022. www.vagrantup.com/docs

Helmond, A. (2020). *DMI Tools*. Zugriff am 05.05.2022. https://wiki.digitalmethods.net/Dmi/ToolDatabase

Herman, I. (W3C Semantic Web, Hrsg.). (2012). *HTML Structured Data Extractor to RDF*. Zugriff am 30.05.2022. https://www.w3.org/2012/sde/

Hickson, I., Pieters, S., van Kesteren, A., Jägenstedt, P. & Denicola, D. (WHATWG, Hrsg.). (2022). *HTML. Living Standard*. Zugriff am 30.05.2022. https://html.spec.whatwg.org/multipage/

Ho, D. (2022). Notepad++ (Version 8.4.1) [Computer software]. https://notepad-plus-plus.org/

Holistics.io. (2022). *dbdiagram.io. Draw Entity-Relationship Diagrams, Painlessly*. Zugriff am 01.06.2022. https://dbdiagram.io/

Holtz, Y. (2018). *The R Graph Gallery*. Zugriff am 24.04.2022. https://r-graph-gallery.com/

Honnibal, M. & Montani, I. (2020). spaCy: Industrial-Strength Natural Language Processing in Python (Version 3.3) [Computer software]: Explosion. https://spacy.io/

Hu, D. J. (2009). *Latent Dirichlet Allocation for Text, Images, and Music,* UCSD CSE Department. Zugriff am 03.04.2022. https://cseweb.ucsd.edu/~dhu/docs/research_exam09.pdf

Hunter, J., Dale, D., Firing, E., Droettboom, M. & Matplotlib development team. (2022). Matplotlib: Visualization with Python (Version 3.5.2) [Computer software]. https://matplotlib.org/

Huynh, D. (2022). OpenRefine (Version 3.5.2) [Computer software]: Metaweb Technologies. https://openrefine.org/

Hydra W3C Community Group. (2018). *API Documentation*. Zugriff am 18.05.2021. http://www.hydra-cg.com/drafts/use-cases/2.api-documentation.md

IBM. (2022). *IBM Watson Studio. Erstellen und skalieren Sie vertrauenswürdige KI in jeder Cloud. Automatisieren Sie den KI-Lebenszyklus für ModelOps.* Zugriff am 30.05.2022. https://www.ibm.com/de-de/cloud/watson-studio

ICA CM. (2017). *About the ICA Computational Methods Interest Group*. Zugriff am 29.11.2019. http://ica-cm.org

Imfernsehen. (2022). *Serien A-Z. R*. Zugriff am 30.05.2022. https://www.fernsehserien.de/serien-a-z/r

IMS. Institut für Maschinelle Sprachverarbeitung, Universität Stuttgart. (2003). *TIGER Korpus 2.2*. Zugriff am 08.05.2022. https://www.ims.uni-stuttgart.de/forschung/ressourcen/korpora/tiger/

Inkscape. (2022). Open Source Scalable Vector Graphics Editor (Version 1.1.2) [Computer software]. https://inkscape.org/

International Movie Database. (2022). *IMDb Datasets*. Zugriff am 23.05.2022. https://www.imdb.com/interfaces/

ISO/IEC 2382. (2015). *Information technology – Vocabulary*.

IVW. (2022). *Informationsgesellschaft zur Feststellung der Verbreitung von Werbeträgern*. Zugriff am 16.05.2022. https://www.ivw.de/

Jacobson, D., Brail, G. & Woods, D. (2012). *APIs: A strategy guide. Creating channels with application programming interfaces.* Beijing: O'Reilly.

Jacomy, M., Venturini, T., Heymann, S. & Bastian, M. (2014). ForceAtlas2, a continuous graph layout algorithm for handy network visualization designed for the Gephi software. *PloS One*, *9*(6), e98679. https://doi.org/10.1371/journal.pone.0098679

James, G., Witten, D., Hastie, T. & Tibshirani, R. (2013). *An Introduction to Statistical Learning. With Applications in R*. New York: Springer. https://doi.org/10.1007/978-1-4614-7138-7

Jannidis, F., Kohle, H. & Rehbein, M. (2017). *Digital Humanities. Eine Einführung*. Stuttgart: J.B. Metzler.

Jansen, D. (2003). *Einführung in die Netzwerkanalyse. Grundlagen, Methoden, Forschungsbeispiele* (2.Aufl.). Wiesbaden: Springer Fachmedien. https://doi.org/10.1007/978-3-663-09875-1

Jeppson, H., Hofmann, H., Di Cook & Wickham, H. (2021). ggmosaic: Mosaic Plots in the 'ggplot2' Framework (Version 0.3.3) [Computer software]. https://cran.r-project.org/package=ggmosaic

JetBrains. (2022a). PyCharm. The Python IDE for Professional Developers (Version 2022.1.3) [Computer software]. www.jetbrains.com/pycharm

JetBrains. (2022b). *The State of Developer Ecosystem 2021*. Zugriff am 16.01.2023. https://www.jetbrains.com/lp/devecosystem-2021/

Johnson, B. & Turner, L. A. (2003). Data Collection Strategies in Mixed Methods Research. In A. Tashakkori & C. Teddlie (Hrsg.), *Handbook of Mixed Methods in Social & Behavioral Research* (S. 297–319). Thousand Oaks: Sage.

JSON-LD. (2022). *JSON-LD Playground*. Zugriff am 30.05.2022. https://json-ld.org/playground/

Jünger, J. (2018). Mapping the field of automated data collection on the web. Data types, collection approaches and their research logic. In C. M. Stützer, M. Welker & M. Egger (Hrsg.), *Computational social science in the age of big data. Concepts, methodologies, tools, and applications* (S. 104–130). Köln: Halem.

Jünger, J. (2020). *Facepager. Wiki*. Zugriff am 03.05.2022. https://github.com/strohne/Facepager/wiki

Jünger, J. (2021). A brief history of APIs. How social media providers shape the opportunities and limitations of online research. In U. Engel, A. Quan-Haase, S. X. Liu &

L. Lyberg (Hrsg.), *Handbook of Computational Social Science* (S. 17–32). London: Routledge. https://doi.org/10.4324/9781003025245-3

Jünger, J. & Keyling, T. (2015). *Facepager commit e570de4, bug fix for downloaded files.* https://github.com/strohne/Facepager/commit/897d000747ec0b4b5d393d0bff86a89a59d617f0#diff-40fe3da5e33280189b65195d7cb0220d

Jünger, J. & Keyling, T. (2022). Facepager. An application for automated data retrieval on the web. (Version 4.4.4) [Computer software]. https://github.com/strohne/Facepager/

Jünger, J. & Schade, H. (2018). Liegt die Zukunft der Kommunikationswissenschaft in der Vergangenheit? Ein Plädoyer für Kontinuität statt Veränderung bei der Analyse von Digitalisierung. *Publizistik*, *63*(4), 497–512. https://doi.org/10.1007/s11616-018-0457-6

Jünger, J., Geise, S. & Hänelt, M. (2022). Unboxing Computational Social Media Research From a Datahermeneutical Perspective: How Do Scholars Address the Tension Between Automation and Interpretation? *International Journal of Communication*, (16), 1482–1505.

Jupyter. (2022). JupyterLab. A Next-Generation Notebook Interface [Computer software]. https://jupyter.org/

Kabacoff, R. I. (2017). *Quick-R*. Zugriff am 24.04.2022. https://www.statmethods.net/

Kaggle. (2013). *Challenges in Representation Learning: Facial Expression Recognition Challenge. Learn facial expressions from an image*. Research Prediction Competition. Zugriff am 08.05.2022. https://www.kaggle.com/c/challenges-in-representation-learning-facial-expression-recognition-challenge/leaderboard

Kaggle. (2022). *Kaggle: Your Machine Learning and Data Science Community*. Zugriff am 23.05.2022. https://www.kaggle.com/

Kearney, M. (2019). rtweet: Collecting and analyzing Twitter data. *The Journal of Open Source Software*, *4*(42), 1829. https://doi.org/10.21105/joss.01829

Khaireddin, Y. & Chen, Z. (2021). *Facial Emotion Recognition: State of the Art Performance on FER2013*. Zugriff am 08.05.2021. http://arxiv.org/pdf/2105.03588v1

Kirby, K. R., Gray, R. D., Greenhill, S. J., Jordan, F. M., Gomes-Ng, S., Bibiko, H.-J. et al. (2016). D-PLACE: A Global Database of Cultural, Linguistic and Environmental Diversity. *PloS one*, *11*(7), e0158391. https://doi.org/10.1371/journal.pone.0158391

Kleene, S. C. (1956). Representation of Events in Nerve Nets and Finite Automata. In C. E. Shannon & J. McCarthy (Hrsg.), *Automata Studies.* (S. 3–42). Princeton: Princeton University Press. https://doi.org/10.1515/9781400882618-002

Kluge, F. (2002). *Etymologisches Wörterbuch der deutschen Sprache* (24., durchges. und erw. Aufl.). Berlin: de Gruyter.

König, L., Pfeiffer-Bohnen, F. & Schmeck, H. (2016). *Theoretische Informatik – ganz praktisch*. Berlin: De Gruyter Oldenbourg. https://doi.org/10.1515/9783110412086

Kotsios, A., Magnani, M., Vega, D., Rossi, L. & Shklovski, I. (2019). An Analysis of the Consequences of the General Data Protection Regulation on Social Network Research. *ACM Transactions on Social Computing*, *2*(3), 1–22. https://doi.org/10.1145/3365524

Krippendorff, K. (2013). *Content analysis. An introduction to its methodology* (3. Aufl.). Los Angeles: Sage.

Krzywinski, M., Birol, I., Jones, S. J. M. & Marra, M. A. (2012). Hive plots. Rational approach to visualizing networks. *Briefings in Bioinformatics*, *13*(5), 627–644. https://doi.org/10.1093/bib/bbr069

Kuchling, A. M. (Python Software Foundation, Hrsg.). (2017). *Regular Expression HOWTO. The Backslash Plague*. Zugriff am 30.05.2022. https://docs.python.org/3.3/howto/regex.html#the-backslash-plague

Kunze, M. (1998). Laßt es leuchten. LAMP: ein datenbankgestütztes Web-Publishing-System mit Freeware. *c't*, (12), 230–231.

Kupietz, M., Belica, C., Keibel, H. & Witt, A. (2010). The German Reference Corpus DeReKo: A Primordial Sample for Linguistic Research. *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC'10)*. 1848–1854. http://www.lrec-conf.org/proceedings/lrec2010/pdf/414_Paper.pdf

Latour, B. (1996). On actor-network theory. A few clarifications. *Soziale Welt*, *47*(4), 369–381. https://www.jstor.org/stable/40878163

Laumann, E. O., Marsden, P. V. & Prensky, D. (1983). The boundary specification problem in network analysis. In R. S. Burt & M. J. Minor (Hrsg.), *Applied network analysis. A methodological introduction* (S. 18–34). Beverly Hills: Sage.

LeCun, Y. & Bengio, Y. (2003). Convolutional networks for images, speech, and time series. In S. Amari (Hrsg.), *The handbook of brain theory and neural networks* (2. Aufl., S. 276–278). Cambridge: MIT Press.

LeCun, Y., Bengio, Y. & Hinton, G. (2015). Deep learning. *Nature*, *521*, 436–444. https://doi.org/10.1038/nature14539

Leetaru, K. H. (2021). *The GDELT Project*. Zugriff am 08.05.2022. https://www.gdeltproject.org/

GESIS. (2022). *GESIS-Suche*. Zugriff am 18.05.2022. https://search.gesis.org/

Le Pennec, E. & Slowikowski, K. (2019). ggwordcloud. A Word Cloud Geom for 'ggplot2' (Version 0.5.0) [Computer software]. https://cran.r-project.org/package=ggwordcloud

Leskovec, J. & Faloutsos, C. (2006). Sampling from large graphs. In L. Ungar, M. Craven, D. Gunopulos & T. Eliassi-Rad (Hrsg.), *KDD '06: Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining* (S. 631–636). New York: Association for Computing Machinery (ACM). https://doi.org/10.1145/1150402.1150479

Lexical Computing. (2022). *Skecth Engine*. https://www.sketchengine.eu/

Link, V., Lohmann, S., Marbach, E., Negru, S. & Wiens, V. (2019). WebVOWL: Web-based Visualization of Ontologies (Version 1.1.7) [Computer software]. http://vowl.visualdataweb.org/webvowl.html

Linvill, D. & Warren, P. (2018). *3 million Russian troll tweets*. Zugriff am 30.05.2022. https://github.com/fivethirtyeight/russian-troll-tweets/

Lu, B., Ott, M., Cardie, C. & Tsou, B. K. (2011). Multi-aspect Sentiment Analysis with Topic Models. In M. Spiliopoulou (Hrsg.), *2011 IEEE 11th International Conference on Data Mining Workshops (ICDMW)* (S. 81–88). Vancouver: IEEE. https://doi.org/10.1109/ICDMW.2011.125

MacroMates. (2021). TextMate (Version 2.0) [Computer software]. https://macromates.com/

Maier, D., Waldherr, A. [A.], Miltner, P., Wiedemann, G., Niekler, A., Keinert, A. et al. (2018). Applying LDA Topic Modeling in Communication Research: Toward a Valid and Reliable Methodology. *Communication Methods and Measures*, *12*(2–3), 93–118. https://doi.org/10.1080/19312458.2018.1430754

Malsch, T. & Schlieder, C. (2004). Communication without Agents? From Agent-Oriented to Communication-Oriented Modeling. In G. Lindemann, D. Moldt & M. Paolucci (Hrsg.), *Regulated Agent-Based Social Systems. First International Workshop, RASTA*

2002, Bologna, Italy, July 16, 2002, Revised Selected and Invited Papers (S. 113–133). Berlin: Springer. https://doi.org/10.1007/978-3-540-25867-4_7

MariaDB Foundation. (2022). MariaDB Server. The open source relational database (Version 10.9.0) [Computer software]. https://mariadb.org/

Markov, A. A. (2006). An Example of Statistical Investigation of the Text Eugene Onegin Concerning the Connection of Samples in Chains. *Science in Context*, *19*(4), 591–600. https://doi.org/10.1017/S0269889706001074

Marres, N. (2017). *Digital sociology. The reinvention of social research*. Cambridge: Polity.

Martinez, A. R. (2012). Part-of-speech tagging. *Wiley Interdisciplinary Reviews: Computational Statistics*, *4*(1), 107–113. https://doi.org/10.1002/wics.195

Matsubara, Y. & GitHub contributors. (2016). *Welcome to PyMySQL's documentation!* Zugriff am 11.07.2022. Verfügbar unter: https://pymysql.readthedocs.io/

McCrae, J. P. (Insight Centre for Data Analytics, Hrsg.). (2021). *The Linked Open Data Cloud*. Zugriff am 01.06.2022. https://lod-cloud.net/

McCulloch, W. S. & Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, *5*(4), 115–133. https://doi.org/10.1007/BF02478259

McPherson, J. (2021). *Debugging with the RStudio IDE*. Zugriff am 24.04.2022. https://support.rstudio.com/hc/en-us/articles/205612627-Debugging-with-RStudio

Media Cloud (2022). *Media Cloud is an open-source platform for media analysis*. Zugriff am 30.05.2022. https://mediacloud.org/

MediaWiki. (2022, 16. Mai). *API:Main page*. Zugriff am 30.05.2022. https://www.mediawiki.org/wiki/API:Main_page

Meier, W. (2003). eXist: An open source native XML database. In G. Goos, J. Hartmanis, J. van Leeuwen, A. B. Chaudhri, M. Jeckle, E. Rahm et al. (Hrsg.), *Web, Web-Services, and Database Systems* (S. 169–183). Berlin: Springer. https://doi.org/10.1007/3-540-36560-5_13

Menge-Sonnentag, R. (2021, 11. März). Nichtrassistische Sprache: Python und GitLab schicken Master aufs Abstellgleis. *Heise Online*. https://www.heise.de/news/Nichtrassistische-Sprache-Python-und-GitLab-schicken-Master-aufs-Abstellgleis-5077505.html

Merten, K. (1995). *Inhaltsanalyse. Einführung in Theorie, Methode und Praxis* (2., verbesserte Aufl.). Wiesbaden: Springer.

Meta. (2021). *CrowdTangle. A tool from Meta to help follow, analyze, and report on what's happening across social media.* Zugriff am 23.05.2022. https://www.crowdtangle.com/

Meta. (2022). *Meta for Developers*. Zugriff am 30.05.2022. https://developers.facebook.com/

Microformats. (2022). Welcome to the mnicroformats wiki! https://microformats.org/wiki/. Zugriff am 30.05.2022. https://microformats.org/wiki/Main_Page

Microsoft. (2022a). *Azure Cognitive Services documentation. Learn how to use ready-made AI services to build intelligent apps, websites, and bots. Develop software that can see, hear, speak, and interpret your user's needs.* Zugriff am 30.05.2022. https://docs.microsoft.com/en-us/azure/cognitive-services/

Microsoft. (2022b). Visual Studio Code (Version 1.67.1) [Computer software]. https://code.visualstudio.com/

Microsoft. (2022c). Windows Subsystem for Linux (WSL) (Version 2) [Computer software]. https://docs.microsoft.com/de-de/windows/wsl/

Middelbos, R. (2020, 24. Februar). *Schmetterling Insekt Blatt Natur*. [Bild]. Zugriff am 03.05.2022. https://pixabay.com/de/photos/schmetterling-insekt-blatt-natur-4873368/

Mimno, D., Wallach, H., Tally, E., Leenders, M. & McCallum, A. (2011). Optimizing Semantic Coherence in Topic Models. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing* (S. 262–272). Edinburgh: Association for Computational Linguistics (ACL).

Ministerium der Justiz Nordrhein-Westfalen. (2020). *Gemeinsames Registerportal der Länder*. Zugriff am 16.05.2022. https://www.handelsregister.de/rp_web/welcome.xhtml

Mitchell, T. M. (1997). *Machine learning*. Boston: WCB/McGraw-Hill.

MongoDB. (2022) (Version 4.0.8) [Computer software]. https://www.mongodb.com/

Monroe, B. L. & Schrodt, P. A. (2008). Introduction to the special issue: The statistical analysis of political text. *Political Analysis*, *16*(4), 351–355. https://doi.org/10.1093/pan/mpn017

Moretti, F. (2013). *Distant reading* (4. Aufl.). London: Verso.

Mozilla (2022). Firefox Browser (Version 99.0.1) [Computer software]. https://www.mozilla.org/de/firefox/new

Müller, K., Wickham, H. & R Special Interest Group on Databases. (2022a). DBI: R Database Interface (Version 1.1.3) [Computer software]. https://cran.r-project.org/package=DBI

Müller, K., Wickham, H., Francois, R. & Bryan, J. (2022b). tibble: Simple Data Frames (Version 3.1.7) [Computer software]. https://cran.r-project.org/package=tibble

Munroe, R. (2013). *Automation. xkdc: A webcomic of romance, sarcasm, math, and language.* Zugriff am 16.05.2022. https://xkcd.com/1319/

Muthukadan, B. (2018). *Selenium with Python. WebDriver API*. Zugriff am 03.07.2022. https://selenium-python.readthedocs.io/api.html

Natale, S. & Ballatore, A. (2020). Imagining the thinking machine: Technological myths and the rise of artificial intelligence. *Convergence: The International Journal of Research into New Media Technologies*, *26*(1), 3–18. https://doi.org/10.1177/1354856517715164

Nederhof, M. & Satta, G. (2010). Theory of Parsing. In A. Clark, C. Fox & S. Lappin (Hrsg.), *The Handbook of Computational Linguistics and Natural Language Processing* (S. 105–130). Oxford: Wiley-Blackwell. https://doi.org/10.1002/9781444324044.ch4

Neo4j. (2022). Neo4j (Version 4.4.7) [Computer software]. https://neo4j.com/

Nepusz, T. (2022). igraph. Network Analysis and Visualization (Version 1.3.2) [Computer software]. https://cran.r-project.org/package=igraph

Neuendorf, K. A. (2017). *The content analysis guidebook* (2. Aufl.). Los Angeles: Sage.

Newman, M. E. J. (2003). Mixing patterns in networks. *Physical Review*, *67*(2), 26126. https://doi.org/10.1103/PhysRevE.67.026126

Newman, M. E. (2005). Power laws, Pareto distributions and Zipf's law. *Contemporary Physics*, *46*(5), 323–351. https://doi.org/10.1080/00107510500052444

NFDI. (2022). *Nationale Forschungsdaten Infrastruktur*. Zugriff am 15.01.2023. https://www.nfdi.de

Ng, A. Y. (coursera, Hrsg.). (2022). *Machine Learning by Stanford University*. Zugriff am 06.05.2022. https://www.coursera.org/learn/machine-learning

Nielsen, J. (1993). *Usability engineering*. Boston: AP Professional.

Nikita, M. & Chaney, N. (2020). ldatuning. Tuning of the Latent Dirichlet Allocation Models Parameters (Version 1.0.2) [Computer software]. https://cran.r-project.org/package=ldatuning

Octopus Data. (2022). Octoparse. Easy Web Scraping for Anyone (Version 8.5.2) [Computer software]. https://www.octoparse.com/

OMG. (2022). *Unified Modeling Language (UML)*. Zugriff am 24.05.2022. https://www.uml.org

Ooms, J. (2014). The jsonlite Package: A Practical and Consistent Mapping Between JSON Data and R Objects. http://arxiv.org/pdf/1403.2805v1

Ooms, J. & McNamara, J. (2021). writexl. Export Data Frames to Excel 'xlsx' Format (Version 1.4.0) [Computer software]. https://cran.r-project.org/package=writexl

Ooms, J., James, D., DebRow, S., Wickham, H., Horner, J. & RStudio. (2021). RMySQL. Database Interface and 'MySQL' Driver for R (Version 0.10.23) [Computer software]. https://cran.r-project.org/package=RMySQL

OpenAI. (2022). *Build next-gen apps with OpenAI's powerful models*. Zugriff am 26.04.2022. https://openai.com/api/

OpenAPI Initiative (2022). *The OpenAPI Specification Repository*. Zugriff am 16.05.2022. https://github.com/OAI/OpenAPI-Specification

OpenCorporates. (2022). *Get data access to over 200 million companies*. Zugriff am 30.05.2022. https://api.opencorporates.com/

OpenLink Software. (2021). *The OpenLink Structured Data Sniffer (OSDS)*. Zugriff am 30.05.2022. https://osds.openlinksw.com/

OpenRefine (2021). *Reconcilable Data Sources*. Zugriff am 18.05.2022. https://github.com/OpenRefine/OpenRefine/wiki/Reconcilable-Data-Sources

OpenLink Software. (2022). *SPARQL Query Editor*. Zugriff am 01.06.2022. https://dbpedia.org/sparql/

Oracle. (2021). MySQL (Version 8.0) [Computer software]. https://www.mysql.com/

Oracle. (2022a). *Get Java for desktop applications*. https://www.java.com/de/download/

Oracle. (2022b). *MySQL Documentation*. Zugriff am 30.05.2022. https://dev.mysql.com/doc/

Oracle. (2022c). VirtualBox (Version 6.1.34) [Computer software]. https://www.virtualbox.org/

Pal, J., Koncz, T., Varkoly, B., Lukacs, P., Kocsis, E. & Teschner, F. (2020). googleCloudVisionR. Access to the 'Google Cloud Vision' API for Image Recognition, OCR and Labeling (Version 0.2.0) [Computer software]. https://cran.r-project.org/package=googleCloudVisionR

Pandas development team. (2022a). Pandas (Version 1.4.3) [Computer software]. *Zenodo*. https://doi.org/10.5281/zenodo.3509134

Pandas development team. (2022b). *DataFrame*. Zugriff am 25.04.2022. https://pandas.pydata.org/pandas-docs/stable/reference/frame.html

Pandas development team. (2022c). *Working with text data*. Zugriff am 25.04.2022. https://pandas.pydata.org/pandas-docs/stable/user_guide/text.html

Pedersen, T. L. (2021). ggraph. An Implementation of Grammar of Graphics for Graphs and Networks (Version 2.0.5) [Computer software]. https://cran.r-project.org/package=ggraph

Pedersen, T. L. (2022). tidygraph. A Tidy API for Graph Manipulation (Version 1.2.1) [Computer software]. https://cran.r-project.org/package=tidygraph

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O. et al. (2011). Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, *12*, 2825–2830.

Peirce, C. S. (1906). Prolegomena to an apology for pragmaticism. *Monist*, *16*(4), 492–546. https://doi.org/10.5840/monist190616436

Pennebaker, J. W., Booth, R. J., Boyd, R. L. & Francis, M. E. (2022). Linguistic Inquiry and Word Count: LIWC-22 (Version 5) [Computer software]: Pennebaker Conglomerates. https://www.liwc.app/

Pennington, J., Socher, R. & Manning, C. (2014). GloVe: Global Vectors for Word Representation. In A. Moschitti, B. Pang & W. Daelemans (Hrsg.), *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)* (S. 1532–1543). Stroudsburg: Association for Computational Linguistics (ACL). https://doi.org/10.3115/v1/D14-1162

Peroni, S. & Daquino, M. (2020). *The unifying REST API for all the OpenCitations Indexes*. Zugriff am 30.05.2022. https://opencitations.net/index/api/v1

Perspective (2021). *Using machine learning to reduce toxicity online. Perspective API can help mitigate toxicity and ensure healthy dialogue online*. Jigsaw. Zugriff am 30.05.2022. https://www.perspectiveapi.com/

Petrou, T. (2017). Selecting Subsets of Data in Pandas. *Medium*. Zugriff am 25.04.2022. https://medium.com/dunder-data/selecting-subsets-of-data-in-pandas-6fcd0170be9c

Piantadosi, S. T. (2014). Zipf's word frequency law in natural language: a critical review and future directions. *Psychonomic Bulletin & Review*, *21*(5), 1112–1130. https://doi.org/10.3758/s13423-014-0585-6.

Pirogov, S. (2022). Webdriver Manager for Python (Version 3.5.4) [Computer software]. https://pypi.org/project/webdriver-manager/

Plotly. (2022). *The front end for ML and data science models*. Zugriff am 03.05.2022. https://plotly.com/

Postman. (2022). Postman (Version 9.16.0) [Computer software]. https://www.postman.com/

Princeton University. (2010). *WordNet. A Lexical Database for English*. Zugriff am 08.05.2022. https://wordnet.princeton.edu/

Pritchard, J. K., Stephens, M. & Donnelly, P. (2000). Inference of population structure using multilocus genotype data. *Genetics*, *155*(2), 945–959. https://doi.org/10.1093/genetics/155.2.945

Puschmann, C. & Haim, M. (2021). *useNews*. [Dataset]. Zugriff am 28.01.2022. https://osf.io/uzca3/

Python Software Foundation. (2019a). *Python Success Stories*. Zugriff am 24.12.2019. https://www.python.org/about/success/ilm/

Python Software Foundation. (2019b). *General Python FAQ*. Zugriff am 24.12.2019. https://docs.python.org/3/faq/general.html#why-is-it-called-python

Python Software Foundation. (2022a). *Lexical analysis. String and Bytes literals*. Zugriff am 30.05.2022. https://docs.python.org/3/reference/lexical_analysis.html#string-and-bytes-literals

Python Software Foundation. (2022b). *Built-in Types. Text Sequence Type*. Zugriff am 25.04.2022. https://docs.python.org/3/library/stdtypes.html#text-sequence-type-str

Python Software Foundation. (2022c). Python (Version 3.10.5) [Computer software]. https://docs.python.org/3/index.html

Python Software Foundation. (2022d). *Virtual Environments and Packages*. Zugriff am 25.04.2022. https://docs.python.org/3/tutorial/venv.html

Python Software Foundation. (2022e). os. Miscellaneous operating system interfaces (Version 3.10.5) [Computer software]. https://docs.python.org/3/library/os.html

Quanteda. (2022). *Topic Models*. Zugriff am 08.05.2022. https://tutorials.quanteda.io/machine-learning/topicmodel/
Quillian, M. R. (1967). Word concepts: a theory and simulation of some basic semantic capabilities. *Behavioral Science*, *12*(5), 410–430. https://doi.org/10.1002/bs.3830120511
R Core Team. (2021). *R Language Definition. Version 4.1.2*. Zugriff am 25.04.2022. https://cran.r-project.org/doc/manuals/r-release/R-lang.pdf
R Core Team. (2022). *The R Project for Statistical Computing. R version 4.2.0*. Zugriff am 24.04.2022. https://www.r-project.org/
Radford, A., Narasimhan, K. S., Salimans, T. & Sutskever, I. (2022). *Improving language understanding by generative pre-training*. Zugriff am 26.04.2022. https://cdn.openai.com/research-covers/language-unsupervised/language_understanding_paper.pdf
RapidMiner. (2021). RapidMiner (Version 9.10.0) [Computer software]. https://rapidminer.com/
RatSWD. (2019). *Big Data in den Sozial-, Verhaltens- und Wirtschaftswissenschaften: Datenzugang und Forschungsdatenmanagement.* Berlin: RatSWD. https://doi.org/10.17620/02671.39
RDFa. (2022). *Linked Data in HTML*. Zugriff am 30.05.2022. http://rdfa.info/
Redis. (2022). Redis (Version 7.0) [Computer software]. https://redis.io/
Reese, A. (2020). *Trump Tweets. Tweets from @realdonaldtrump*. [Dataset], kaggle.com. Zugriff am 30.05.2022. https://www.kaggle.com/datasets/austinreese/trump-tweets
Reitz, K. (2021). *The Hitchhiker's Guide to Python*. Zugriff am 25.04.2022. https://docs.python-guide.org/
Reitz, K. (2022). *Requests: HTTP for Humans. The User Guide*. Zugriff am 03.05.2022. https://requests.readthedocs.io/
Ricci, F., Rokach, L. & Shapira, B. (Hrsg.). (2015). *Recommender Systems Handbook*. Boston: Springer. https://doi.org/10.1007/978-1-4899-7637-6
Richardson, L. (2020). *Beautiful Soup Documentation*. Zugriff am 03.05.2022. https://www.crummy.com/software/BeautifulSoup/bs4/doc/
Richardson, L. (2022). Beautiful Soup [Computer software]. https://www.crummy.com/software/BeautifulSoup
Rieger, J. (2021). ldaPrototype. Prototype of Multiple Latent Dirichlet Allocation Runs (Version 0.3.1) [Computer software]. https://cran.r-project.org/package=ldaPrototype
Rieger, J., Rahnenführer, J. & Jentsch, C. (2020). Improving Latent Dirichlet Allocation: On Reliability of the Novel Method LDAPrototype. In E. Métais, F. Meziane, H. Horacek & P. Cimiano (Hrsg.), *Natural Language Processing and Information Systems* (S. 118–125). Cham: Springer. https://doi.org/10.1007/978-3-030-51310-8_11
Rob, C. & Singh, L. (2022). The Evolution of Topic Modeling. ACM Computing Surveys *54*(10s), 1–35 https://doi.org/10.1145/3507900
Roberts, M. E., Stewart, B. M. & Airoldi, E. M. (2016). A Model of Text for Experimentation in the Social Sciences. *Journal of the American Statistical Association*, *111*(515), 988–1003. https://doi.org/10.1080/01621459.2016.1141684
Robins, G., Pattison, P., Kalish, Y. & Lusher, D. (2007). An introduction to exponential random graph (p*) models for social networks. *Social Networks*, *29*(2), 173–191. https://doi.org/10.1016/j.socnet.2006.08.002
Robinson, D., Bryan, J. & Elias, J. (2020). fuzzyjoin: Join Tables Together on Inexact Matching (Version 0.1.6) [Computer software]. https://cran.r-project.org/package=fuzzyjoin

Robinson, D., Misra, K. & Silge, J. (2021). widyr. Widen, Process, then Re-Tidy Data (Version 0.1.4) [Computer software]. https://cran.r-project.org/package=widyr

Roche, X. (2017). Httrack. Website Copier (Version 3.49-2) [Computer software]. https://www.httrack.com/

Roesslein, J. (2020). Tweepy. Twitter for Python! [Computer software]. https://github.com/tweepy/tweepy

Rogers, R. (2010). Internet Research. The Question of Method. A Keynote Address from the YouTube and the 2008 Election Cycle in the United States Conference. *Journal of Information Technology & Politics*, *7*(2–3), 241–260. https://doi.org/10.1080/19331681003753438

Rosenblatt, F. (1958). The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological Review*, *65*(6), 386–408. https://doi.org/10.1037/h0042519

Rossi, L. & Ahmed, N. K. (2022). *Network Repository. A Scientific Network Data Repository with Interactive Visualization and Mining Tools*. Zugriff am 30.05.2022. https://networkrepository.com/

Rowley, J. (2007). The wisdom hierarchy: representations of the DIKW hierarchy. *Journal of Information Science*, *33*(2), 163–180. https://doi.org/10.1177/0165551506070706

RStudio. (2020). *Shinyapps.io*. Zugriff am 03.05.2022. https://www.shinyapps.io/

RStudio. (2022a). *Download the RStudio IDE*. Zugriff am 25.04.2022. https://www.rstudio.com/products/rstudio/download/

RStudio (2022b). *RStudio Cheatsheets*. Zugriff am 24.04.2022. https://www.rstudio.com/resources/cheatsheets/

Rumelhart, D. E., Hinton, G. E. & Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature*, *323*(6088), 533–536. https://doi.org/10.1038/323533a0

Russell, S. J. & Norvig, P. (2012). *Künstliche Intelligenz. Ein moderner Ansatz* (3., aktual. Aufl.). München: Pearson.

Ruttenberg, A. (2020). *Basic Formal Ontology (BFO)*. http://basic-formal-ontology.org/

Sack, H. & Koutraki, M. (2017). *Information Service Engineering,* openHPI. https://open.hpi.de/courses/semanticweb2017/

Salamanos, N., Voudigari, E. & Yannakoudakis, E. J. (2017). Deterministic graph exploration for efficient graph sampling. *Social Network Analysis and Mining*, *7*, 24. https://doi.org/10.1007/s13278-017-0441-6

Sambare, M. (2020). *FER-2013. Learn facial expressions from an image*. [Dataset], kaggle.com. Zugriff am 08.05.2022. https://www.kaggle.com/datasets/msambare/fer2013

Samuel, A. L. (1959). Some studies in machine learning using the game of checkers. *IBM Journal*, *3*(3), 210–229. https://doi.org/10.1147/rd.33.0210

Santacroce, F. (2015). *Git essentials. Create, merge, and distribute code with Git, the most powerful and flexible versioning system available*. Birmingham: Packt Publishing.

Scharkow, M. (2011). Zur Verknüpfung manueller und automatischer Inhaltsanalyse durch maschinelles Lernen. *Medien & Kommunikationswissenschaft*, *59*(4), 545–562. https://doi.org/10.5771/1615-634x-2011-4-545

SchedMD. (2021). *Slurm workload manager. Documentation*. Zugriff am 03.05.2022. https://slurm.schedmd.com/

Schindler, D., Bensmann, F., Dietze, S. & Krüger, F. (2021). *SoMeSci – A 5 Star Open Data Gold Standard Knowledge Graph of Software Mentions in Scientific Articles*. Zugriff am 30.05.2022. https://data.gesis.org/somesci/

Schler, J., Koppel, M., Argamon, S. & Pennebaker, J. (2005). Effects of Age and Gender on Blogging. *Proceedings of 2006 AAAI Spring Symposium on Computational Approaches for Analyzing Weblogs*. http://www.cs.biu.ac.il/~schlerj/schler_springsymp06.pdf

Schubert, L. (2020). Computational Linguistics. In E. N. Zalta (Hrsg.), *The Stanford Encyclopedia of Philosophy*. Zugriff am 01.04.2020. https://plato.stanford.edu/archives/spr2020/entries/computational-linguistics/

Schultz, F., Kleinnijenhuis, J., Oegema, D., Utz, S. & van Atteveldt, W. (2012). Strategic framing in the BP crisis. A semantic network analysis of associative frames. *Public Relations Review*, *38*(1), 97–107. https://doi.org/10.1016/j.pubrev.2011.08.003

Scikit-learn developers. (2022). *sklearn.neural_network.MLPClassifier*. Zugriff am 08.05.2022. https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html

ScrapeBox. (2022). ScrapeBox (Version 2.0) [Computer software]. http://www.scrapebox.com/

Selfhtml. (2021). *XML/XSL/XPath*. Zugriff am 30.05.2022. https://wiki.selfhtml.org/wiki/XML/XSL/XPath

Selfhtml. (2022). *CSS/Tutorials/Selektoren*. Zugriff am 30.05.2022. https://wiki.selfhtml.org/wiki/CSS/Tutorials/Selektoren

Semantalytics. (2022). *Awesome Semantic Web. A curated list of various semantic web and linked data resources.* Zugriff am 30.05.2022. https://github.com/semantalytics/awesome-semantic-web

SemEval. (2022). *International Workshop on Semantic Evaluation*. Zugriff am 30.05.2022. https://semeval.github.io/

Shah, D. V., Cappella, J. N. & Neuman, W. R. (2015). Big Data, Digital Media, and Computational Social Science. *The ANNALS of the American Academy of Political and Social Science*, *659*(1), 6–13. https://doi.org/10.1177/0002716215572084

Shannon, P., Markiel, A., Ozier, O., Baliga, N. S., Wang, J. T., Ramage, D. et al. (2003). Cytoscape: a software environment for integrated models of biomolecular interaction networks. *Genome Research*, *13*(11), 2498–2504. https://doi.org/10.1101/gr.1239303

Sievert, C. & Shirley, K. E. (2014). LDAvis: A method for visualizing and interpreting topics. In *Proceedings of the Workshop on Interactive Language Learning, Visualization, and Interfaces* (S. 63–70). Baltimore: Association for Computational Linguistics (ACL). https://doi.org/10.13140/2.1.1394.3043

Silge, J. & Robinson, D. (2016). tidytext: Text Mining and Analysis Using Tidy Data Principles in R. *The Journal of Open Source Software*, *1*(3), 37. https://doi.org/10.21105/joss.00037

Silge, J. & Robinson, D. (2017). *Text mining with R. A tidy approach*. Sebastopol: O'Reilly.

Silva, V. (2008). *What is the best regular expression to check if a string is a valid URL?,* Stack Overflow. Zugriff am 30.05.2022. https://stackoverflow.com/questions/161738/what-is-the-best-regular-expression-to-check-if-a-string-is-a-valid-url

Skinner, G. (2021). *RegExr. RegExr is an online tool to learn, build, & test Regular Expressions*. Zugriff am 30.05.2022. https://regexr.com/

Smith, R. & Tesseract Development Team. (2022). Tesseract [Computer software]. https://github.com/tesseract-ocr

Snijders, T. A. B., Ripley, R., Steglich, C., Koskinen, J., Niezink, N., Amati, V. et al. (2021). RSiena. Siena – Simulation Investigation for Empirical Network Analysis (Version 1.3) [Computer software]. https://cran.r-project.org/package=RSiena

Social Media Data Stewardship. (2021). *Social Media Research Toolkit*. Zugriff am 16.05.2022. https://socialmediadata.org/social-media-research-toolkit/

Solid IT. (2022). *DB-Engines. Informationen zu relationalen und NoSQL Datenbankmanagementsystemen*. Zugriff am 24.05.2022. https://db-engines.com/

Spinu, V., Grolemund, G., Wickham, H., Vaughan, D., Lyttle, I., Costigan, I. et al. (2021). lubridate: Make Dealing with Dates a Little Easier (Version 1.8.0) [Computer software]. https://cran.r-project.org/package=lubridate

Spyder project contributors. (2022). Spyder. The Scientific Python Development Environment (Version 531) [Computer software]. https://www.spyder-ide.org/

SQLite Development Team. (2022). SQLite [Computer software]. https://www.sqlite.org/

Stack Overflow. (2022). *A public platform building the definitive collection of coding questions & answers*. Zugriff am 24.04.2022. https://stackoverflow.com

Statistical Cybermetrics Research Group. (2016). SoSciBot. Web crawler and link analyser for the social sciences and humanities (Version 4) [Computer software]: University of Wolverhampton. http://socscibot.wlv.ac.uk/

Stenberg, D. (2022). curl. A command line tool and library for transferring data with URL syntax [Computer software]. https://curl.se/

Stone, P. J., Dunphy, D., Smith, M. S. & Ogilvie, D. M. (1966). *The General Inquirer. A Computer Approach to Content Analysis*. Cambridge: MIT Press.

Syncro Soft. (2022). Oxygen [Computer software]. https://www.oxygenxml.com/

Taylor & Francis. (2022). *Media, Cultural & Communication Studies Title List 2019*. https://librarianresources.taylorandfrancis.com/collection/media-cultural-communication-studies-collection/media-cult-com-stu/

Telegram. (2020). *MTProto Mobile Protocol*. Zugriff am 28.06.2020. https://core.telegram.org/mtproto

Terras, M., Nyhan, J. & Vanhoutte, E. (Hrsg.). (2013). *Defining digital humanities. A reader*. Farnham: Ashgate.

Text Encoding Initiative. (2022). *Text Encoding Initiative*. Zugriff am 24.05.2022. https://tei-c.org/

The PHP Group. (2022). PHP (Version 8.1.5) [Computer software]. https://www.php.net/

The pip developers. (2022). pip. The package installer for Python [Computer software]. https://pip.pypa.io/

Thelwall, M. [Michael]. (2009). *Introduction to Webometrics. Quantitative Web Research for the Social Sciences*. San Rafael: Morgan & Claypool. https://doi.org/10.2200/S00176ED1V01Y200903ICR004

Thelwall, M. [Mike] & Stuart, D. (2006). Web crawling ethics revisited: Cost, privacy, and denial of service. *Journal of the American Society for Information Science and Technology*, *57*(13), 1771–1779. https://doi.org/10.1002/asi.20388

Thompson, K. (1968). Programming Techniques: Regular expression search algorithm. *Communications of the ACM*, *11*(6), 419–422. https://doi.org/10.1145/363347.363387

ThoughtWorks. (2022). Selenium [Computer software]. https://www.selenium.dev/

Turney, P. D. (2000). Learning Algorithms for Keyphrase Extraction. *Information Retrieval*, *2*(4), 303–336. https://doi.org/10.1023/A:1009976227802

Turney, P. D. & Pantel, P. (2010). From Frequency to Meaning: Vector Space Models of Semantics. *Journal of Artificial Intelligence Research*, *37*, 141–188. https://doi.org/10.1613/jair.2934

Twitter. (2020). *Developer platform*. Zugriff am 30.05.2022. https://developer.twitter.com/en/apply-for-access

Twitter. (2022a). *Follow, search, and get users. API reference GET followers/ids*. Zugriff am 03.05.2022. https://developer.twitter.com/en/docs/twitter-api/v1/accounts-and-users/follow-search-get-users/api-reference/get-followers-ids

Twitter. (2022b). *Get Tweet timelines. API reference GET statuses/user_timeline*. Zugriff am 04.05.2022. https://developer.twitter.com/en/docs/twitter-api/v1/tweets/timelines/api-reference/get-statuses-user_timeline

Twitter. (2022c). *Platform overview. Get started with the Twitter Developer Platform*. Zugriff am 03.05.2022. https://developer.twitter.com/en/docs/platform-overview

Twitter. (2022d). *Twitter Developer Platform. Use Cases, Tutorials, & Documentation*. Zugriff am 30.05.2022. https://developer.twitter.com/en

Twitter. (2022e). *Users lookup. API reference GET /2/users/by/username/:username*. Zugriff am 03.05.2022. https://developer.twitter.com/en/docs/twitter-api/users/lookup/api-reference/get-users-by-username-username

Twitter. (2022f). *Follow, search, and get users. API reference GET followers/list*. Zugriff am 03.05.2022. https://developer.twitter.com/en/docs/twitter-api/v1/accounts-and-users/follow-search-get-users/api-reference/get-followers-list

Unicode. (2021). *Submitting Emoji Proposals*. Zugriff am 24.05.2022. https://unicode.org/emoji/proposals.html

Universität Münster. (2022). *High Performance Computing*. Zugriff am 03.05.2022. https://www.uni-muenster.de/IT/services/unterstuetzungsleistung/hpc/

University of Michigan. (2022). *ICPSR: Inter-University Consortium for Political and Social Research*. Zugriff am 30.05.2022. https://www.icpsr.umich.edu/web/pages/

University of Oxford. (2021). *OTA: Oxford Text Archive. A repository of full-text literary and linguistic resources. Thousands of texts in more than 25 languages*. Zugriff am 30.05.2022. https://ota.bodleian.ox.ac.uk/repository/xmlui/

Uppsala Conflict Data Program. (2016). *Georeferenced Event Dataset (GED). Global version 17.1 [Data set]*. Zugriff am 30.05.2022. http://ucdp.uu.se/downloads/

Van Atteveldt, W. (2008). *Semantic network analysis. Techniques for extracting, representing and querying media content*. Charleston: BookSurge.

Van Atteveldt, W. & Peng, T.-Q. (2018). When Communication Meets Computation: Opportunities, Challenges, and Pitfalls in Computational Communication Science. *Communication Methods and Measures*, *12*(2–3), 81–92. https://doi.org/10.1080/19312458.2018.1458084

Van Atteveldt, W. & Welbers, K. (2022). Rsyntax. R library to help dealing with syntactic structure [Computer software]. https://github.com/vanatteveldt/rsyntax

Van Atteveldt, W., van der Velden, M. A. C. G. & Boukes, M. (2021). The Validity of Sentiment Analysis: Comparing Manual Annotation, Crowd-Coding, Dictionary Approaches, and Machine Learning Algorithms. *Communication Methods and Measures*, *15*(2), 121–140. https://doi.org/10.1080/19312458.2020.1869198

Van Atteveldt, W., Sheafer, T., Shenhav, S. R. & Fogel-Dror, Y. (2017). Clause Analysis. Using Syntactic Information to Automatically Extract Source, Subject, and Predicate from Texts with an Application to the 2008–2009 Gaza War. *Political Analysis*, *25*(2), 207–222. https://doi.org/10.1017/pan.2016.12

Van Dijck, J. (2020). Seeing the forest for the trees: Visualizing platformization and its governance. *New Media & Society*, *23*(9), 2801–2819. https://doi.org/10.1177/1461444820940293

Van Rossum, G., Warsaw, B. & Coghlan, N. (2013). *PEP 8 – Style Guide for Python Code*. Zugriff am 25.04.2022. https://peps.python.org/pep-0008/

Varoquaux, G. & Joblib developers. (2021). Joblib: running Python functions as pipeline jobs (Version 1.1.0) [Computer software]. https://joblib.readthedocs.io/

Vaughan, D. & Dancho, M. (2022). *furrr: Apply Mapping Functions in Parallel using Futures*. Zugriff am 18.01.2023. https://furrr.futureverse.org/

VERBI Software. (2021). MAXQDA (Version 2022) [Computer software]. https://www.maxqda.com/

Vidgen, B. & Derczynski, L. (2020). Directions in abusive language training data, a systematic review: Garbage in, garbage out. *PloS One*, *15*(12), e0243300. https://doi.org/10.1371/journal.pone.0243300

Visual Geometry Group. (2022). *VoxCeleb. A large scale audio-visual dataset of human speech*. Zugriff am 30.05.2022. https://www.robots.ox.ac.uk/~vgg/data/voxceleb/

Voss, C., Cammarata, N., Goh, G., Petrov, M., Schubert, L., Egan, B. et al. (2021). Visualizing Weights. *Distill*, *6*(2). https://doi.org/10.23915/distill.00024.007

W3C. (2013). *Extensible Markup Language (XML) 1.0 (Fifth Edition). W3C Recommendation*. Zugriff am 24.05.2022. https://www.w3.org/TR/xml/

W3C. (2014). *RDF 1.1 Primer. W3C Working Group Note*. Zugriff am 24.05.2022. https://www.w3.org/TR/rdf11-primer/

W3C. (2015). *RDFa 1.1 Primer – Third Edition. Rich Structured Data Markup for Web Documents. W3C Working Group Note*. Zugriff am 30.05.2022. https://www.w3.org/TR/xhtml-rdfa-primer/

W3C. (2019). *Semantic Web*. Zugriff am 30.05.2022. https://www.w3.org/2001/sw/wiki/Main_Page

W3C. (2021). *Search engines*. Zugriff am 24.05.2022. https://www.w3.org/wiki/Search_engines

W3Schools. (2022a). *HTML Element Reference*. Zugriff am 24.05.2022. https://www.w3schools.com/tags/default.asp

W3Schools. (2022b). *Python Classes and Objects*. Zugriff am 25.04.2022. https://www.w3schools.com/python/python_classes.asp

W3Schools. (2022c). *Python Try Except*. Zugriff am 25.04.2022. https://www.w3schools.com/python/python_try_except.asp

W3Schools. (2022d). *SQL Tutorial*. Zugriff am 30.05.2022. https://www.w3schools.com/sql/

W3Techs. (2022). *Usage statistics of PHP for websites*. Zugriff am 05.07.2022. w3techs.com/technologies/details/pl-php

Waldherr, A., Hilbert, M. & González-Bailón, S. (2021). Worlds of Agents: Prospects of Agent-Based Modeling for Communication Research. *Communication Methods and Measures*, *15*(4), 243–254. https://doi.org/10.1080/19312458.2021.1986478

Wallach, H. M. (2006). Topic modeling: beyond bag-of-words. In W. Cohen & A. Moore (Hrsg.), *Proceedings of the 23rd international conference on Machine learning – ICML ‘06* (S. 977–984). New York: ACM Press. https://doi.org/10.1145/1143844.1143967

Waring, E., Quinn, M., McNamara, A., Arino de la Rubia, E, Zhu, H. & Ellis, S. (2022). skimr. Compact and Flexible Summaries of Data [Computer software]. https://docs.ropensci.org/skimr/

Waskom, M. L. (2021). seaborn: statistical data visualization. *The Journal of Open Source Software*, *6*(60), 3021. https://doi.org/10.21105/joss.03021

Wasserman, S. & Faust, K. (1994). *Social network analysis. Methods and applications*. Cambridge: Cambridge University Press.
Web Scraper. (2021). WebScraper. Making web data extraction easy and accessible for everyone (Version 0.6.4) [Computer software]. https://webscraper.io/
Weiboscope. (2022). *HKU JMSC Weibo Censorship Index*. Journalism and Media Studies Centre. Zugriff am 23.05.2022. https://weiboscope.jmsc.hku.hk/wsr/
Welker, M. (2019). Computer- und onlinegestützte Methoden für die Untersuchung digitaler Kommunikation. In W. Schweiger & K. Beck (Hrsg.), *Handbuch Online-Kommunikation* (S. 531–572). Wiesbaden: Springer Fachmedien. https://doi.org/10.1007/978-3-658-18016-4_21
White, H. C. (2008). *Identity and Control. How Social Formations Emerge* (2. Aufl.). Princeton: Princeton University Press.
Wick, M. (Unxos GmbH, Hrsg.). (2022). *GeoNames*. Zugriff am 24.05.2022. https://www.geonames.org/
Wickham, H. (2014). Tidy Data. *Journal of Statistical Software*, *59*(10). https://doi.org/10.18637/jss.v059.i10
Wickham, H. (2019a). stringr. Simple, Consistent Wrappers for Common String Operations (Version 1.4.0) [Computer software]. https://cran.r-project.org/package=stringr
Wickham, H. (2019b). *Tidyverse. R packages for data science*. Zugriff am 24.04.2022. https://www.tidyverse.org/
Wickham, H. (2021). *The tidyverse style guide*. Zugriff am 24.04.2022. https://style.tidyverse.org/
Wickham, H. (2022a). *httr: Tools for Working with URLs and HTTP*. Zugriff am 05.05.2022. https://httr.r-lib.org/
Wickham, H. (2022b). rvest. Easily Harvest (Scrape) Web Pages [Computer software]. https://rvest.tidyverse.org/
Wickham, H. & Girlich, M. (2022). tidyr: Tidy Messy Data (Version 1.2.0) [Computer software]. https://cran.r-project.org/package=tidyr
Wickham, H. & Grolemund, G. (2016). *Data visualisation*. Zugriff am 24.04.2022. https://r4ds.had.co.nz/data-visualisation.html
Wickham, H., Girlich, M. & Ruiz, E. (2022). dbplyr. A 'dplyr' back end for databases [Computer software]. https://dbplyr.tidyverse.org/
Wickham, H., Miller, E. & Smith, D. (2022). haven: Import and Export 'SPSS', 'Stata' and 'SAS' Files (Version 2.5.0) [Computer software]. https://cran.r-project.org/package=haven
Wickham, H., François, R., Henry, L. & Müller, K. (2022a). dplyr: A Grammar of Data Manipulation (Version 1.0.9) [Computer software]. https://cran.r-project.org/package=dplyr
Wickham, H., Bryan, J., Kalicinski, M., Valery, K., Leitienne, C., Colbert, B. et al. (2022b). readxl: Read Excel Files (Version 1.4.0) [Computer software]. https://cran.r-project.org/package=readxl
Wickham, H., Chang, W., Henry, L., Lin Pederson, T., Takahashi, K., Wilke, C. et al. (2022c). *Function reference ggplot2*. Zugriff am 24.04.2022. https://ggplot2.tidyverse.org/reference/
Wickham, H., Chang, W., Henry, L., Lin Pederson, T., Takahashi, K., Wilke, C. et al. (2022d). *Learning ggplot2*. Zugriff am 24.04.2022. https://ggplot2.tidyverse.org/#learning-ggplot2
Wickham, H., Hester, J., Francois, R., Bryan, J., Bearrows, S., Jylänki, J. et al. (2022e). readr: Read Rectangular Text Data (Version 2.1.2) [Computer software]. https://cran.r-project.org/package=readr
Wikimedia Deutschland. (2022). *Wikipedia. Die freie Enzyklopädie*. Zugriff am 16.05.2022. https://www.wikipedia.de/

Wikipedia. (2021, 8. Oktober). *Liste auflagenstärkster Zeitschriften. Deutschland*. Zugriff am 03.05.2022. https://de.wikipedia.org/wiki/Liste_auflagenst%C3%A4rkster_Zeitschriften#-Deutschland

Wikipedia. (2022a). *List of datasets for machine-learning research*. https://en.wikipedia.org/wiki/List_of_datasets_for_machine-learning_research

Wikipedia. (2022b). *Sparse matrix*. Zugriff am 30.05.2022. https://en.wikipedia.org/wiki/Sparse_matrix

Wilensky, U. (2021). NetLogo (Version 6.2.1) [Computer software]: Center for Connected Learning and Computer-Based Modeling, Northwestern University, Evanston, IL. https://ccl.northwestern.edu/netlogo/6.2.1/

Wilkinson, L. (1999). *The grammar of graphics*. New York: Springer.

Wilkinson, M. D., Dumontier, M., Aalbersberg, I. J. J., Appleton, G., Axton, M., Baak, A. et al. (2016). The FAIR Guiding Principles for scientific data management and stewardship. *Scientific Data*, *3*, 1–9. https://doi.org/10.1038/sdata.2016.18

WinPython development team. (2022). WinPython [Computer software]. https://winpython.github.io/

Wolf, M., Horn, A. B., Mehl, M. R., Haug, S., Pennebaker, J. W. & Kordy, H. (2008). Computergestützte quantitative Textanalyse. *Diagnostica*, *54*(2), 85–98. https://doi.org/10.1026/0012-1924.54.2.85

Wrobel, S., Morik, K. & Joachims, T. (2003). Kapitel 14: Maschinelles Lernen und Data Mining. In G. Görz & J. Schneeberger (Hrsg.), *Handbuch der Künstlichen Intelligenz* (S. 517–597). München: Oldenbourg. https://doi.org/10.1524/9783486598834.517

Zubiaga, A. (2018a). A longitudinal assessment of the persistence of twitter datasets. *Journal of the Association for Information Science and Technology*, *69*(8), 974–984. https://doi.org/10.1002/asi.24026

Zubiaga, A. (2018b). *Twitter event datasets 2012–2016*. [Dataset]. Zugriff am 30.05.2022. https://doi.org/10.6084/m9.figshare.5100460.v2

Zyte. (2022). Scrapy (Version 2.6.1) [Computer software]. https://scrapy.org/